REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY) 13-11-2010	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 1-Aug-2006 - 31-Jul-2010

4. TITLE AND SUBTITLE Efficient Effects-Based Military Planning Final Report	5a. CONTRACT NUMBER W911NF-06-1-0331
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 611102
6. AUTHORS Qiang Ji	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES Rensselaer Polytechnic Institute, Office of Sponsored Research, Rensselaer Polytechnic Institute, Troy, NY 12180 -	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office, P.O. Box 12211, Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) 48134-NS.1

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
This research focused on developing a mathematical framework and the associated methods for effects-based military plan modeling and evaluation. Specifically, we first introduced a unified probabilistic framework based on the Dynamic Influence Diagram to systematically represent the causal relationships between actions and their effects, their interactions, their uncertainties, and their dynamics. We then developed advanced machine learning methods to automatically construct such a model for a given campagin and to learn the model parameters from both

15. SUBJECT TERMS
Military Planning, Influence Diagrams, EBO

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Qiang Ji
UU	UU	UU	UU		19b. TELEPHONE NUMBER 518-276-6440

Report Title

Efficient Effects-Based Military Planning Final Report

ABSTRACT

This research focused on developing a mathematical framework and the associated methods for effects-based military plan modeling and evaluation. Specifically, we first introduced a unified probabilistic framework based on the Dynamic Influence Diagram to systematically represent the causal relationships between actions and their effects, their interactions, their uncertainties, and their dynamics. We then developed advanced machine learning methods to automatically construct such a model for a given campagin and to learn the model parameters from both the training data and the available qualitative domain knowledge. Given the framework, we developed efficient inference methods to quickly evaluate each plan to identify the opimtal plan. This research also includes the development of a user-friendly prototype software to evaluate the performance of the proposed methods for military plan modeling and evaluation. The software can produce a model for a military campaign, automatically learn the model structure and parameters, evaluate various possible plans, and identify the one that best meets the campaign objective. Further information is available at the project's website at http://www.ecse.rpi.edu/~cvrl/EBO/ebo.htm

List of papers submitted or published that acknowledge ARO support during this reporting period. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

1. Cassio de Campos and Qiang Ji, Efficient Structure Learning of Bayesian Networks using Constraints, accepted by Journal of Machine Learning Research.

2. Yongmian Zhang and Qiang Ji, Efficient Sensor Selection for Active Information Fusion, IEEE Transactions on Systems, Man, Cybernetics, Part B, Volume: 40 Issue 3, pages 719 - 728, June,
2010.

3. Wenhui Liao and Qiang Ji, Learning Bayesian Network Parameters Under Incomplete Data with Qualitative Domain Knowledge, Pattern Recognition, Volume 42 , Issue 11, Pages 3046-3056, 2009

4. Wenhui Liao and Qiang Ji, Efficient Non-myopic Value-of-Information Computation for Influence Diagrams, International Journal on Approximate Reasoning, vol. 49, no. 2, pp. 436-450, 2008

5. Weihong Zhang and Qiang Ji, A Factorization Approach To Evaluating Simultaneous Influence Diagrams, IEEE Transactions on Systems, Man, and Cybernetics A, p746-754, Vol. 36, No. 4, July, 2006

Number of Papers published in peer-reviewed journals: 5.00

(b) Papers published in non-peer-reviewed journals or in conference proceedings (N/A for none)

Number of Papers published in non peer-reviewed journals: 0.00

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts): 0

Peer-Reviewed Conference Proceeding publications (other than abstracts):

1. Cassio de Campos and Qiang Ji, Properties of Bayesian Dirichlet scores to learn Bayesian network structures, AAAI, 2010

2. Cassio de Campos, Zhi Zeng, Qiang Ji, An Improved Structural EM to Learn Dynamic Bayesian Nets,International Conference on Pattern Recognition, 2010.

3. Yue Zhao and Qiang Ji, Non-myopic Active Learning with Mutual Information, IEEE International Conference on Automation and Logistics August 16–20, Hong Kong, China, 2010.

4. Yue Zhao and Qiang Ji, An Active Learning Method under Very Limited Initial Labeled Data, IEEE International Conference on Automation and Logistics August 16–20, Hong Kong, China, 2010.

5. Cassio de Campos, Zhi Zeng, and Qiang Ji, Structure Learning of Bayesian Networks using Constraints, International Conference on Machine Learning (ICML), 2009.

6. Cassio de Campos and Qiang Ji, Strategy Selection in Influence Diagrams using Imprecise Probabilities, the 24th Conference on Uncertainty in Artificial Intelligence (UAI), 2008.

7. Yan Tong and Qiang Ji, Learning Bayesian Networks with Qualitative Constraints, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.

8. Cassio de Campos and Qiang Ji, Improving Bayesian Network Parameter Learning using Constraints, International Conference in Pattern Recognition (ICPR), 2008.

9. Wenhui Liao and Qiang Ji, Exploiting Qualitative Domain Knowledge for Learning Bayesian Network Parameters with Incomplete Data, International Conference in Pattern Recognition (ICPR), 2008.

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts): 9

(d) Manuscripts

1) Cassio P. de Campos and Qiang J, Learning Limited Memory In°uence Diagrams, Techncial Report ISL-EBO-2009-01, 2009.

2) Geng Li, Validation of Cassio's software, Techncial Report, ISL-EBO-2008-01.

3) Qiang Ji, Modeling and Evaluating EBO-based Miltiary Planning using the Influence Diagram, Technical report, ISL-EBO-2007-01,2007

4) Cassio P. de Campos, Learning Influence Diagram Parameters using
Convex Optimization for EBO-based Planning, Technical Report, ISL-EBO-2007-02, 2007.

5) Yue Zeng, Exploiting Qualitative Constraints for Learning
Bayesian Networks under Insufficient Data, Technical Report, ISL-EBO-2007-03, 2007.

Number of Manuscripts: 0.00

Patents Submitted

Patents Awarded

Awards

Research Excellence Award, School of Engineering, RPI, 2006

RPI Trustee's Faculty Achievement Award, 2008, 2009

PI Inventor Award, 2008

First place award, Data analysis Competition, International Conference on Machine learning for Signal Processing, 2008.

Graduate Students

NAME	PERCENT_SUPPORTED
Geng Li	0.50
Wenhui Liao	0.50
FTE Equivalent:	**1.00**
Total Number:	**2**

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
Cassio de Campos	1.00
Yonmian Zhang	1.00
Yue Zhao	0.50
FTE Equivalent:	**2.50**
Total Number:	**3**

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Qiang Ji	0.15	No
FTE Equivalent:	**0.15**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	0.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:......	0.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):......	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:......	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:......	0.00

Names of Personnel receiving masters degrees

NAME	
Total Number:	

Names of personnel receiving PHDs

NAME	
Wenhui Liao	
Total Number:	**1**

Names of other research staff

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Journal of Machine Learning Research – (2010) – Submitted 05/10; Published –/–

Efficient Structure Learning of Bayesian Networks using Constraints

Cassio P. de Campos CASSIOPC@ACM.ORG
Dalle Molle Institute for Artificial Intelligence
Galleria 2, Manno 6928, Switzerland

Qiang Ji JIQ@RPI.EDU
Dept. of Electrical, Computer & Systems Engineering
Rensselaer Polytechnic Institute
110 8th Street, Troy, NY 12180, USA

Editor: David Maxwell Chickering

Running title: Efficient Structure Learning of Bayesian Nets

Abstract

This paper addresses the problem of learning Bayesian network structures from data based on score functions that are decomposable. It describes properties that strongly reduce the time and memory costs of many known methods without losing global optimality guarantees. These properties are derived for different score criteria such as Minimum Description Length (or Bayesian Information Criterion), Akaike Information Criterion and Bayesian Dirichlet Criterion. Then a branch-and-bound algorithm is presented that integrates structural constraints with data in a way to guarantee global optimality. As an example, structural constraints are used to map the problem of structure learning in Dynamic Bayesian networks into a corresponding augmented Bayesian network. Finally, we show empirically that the new algorithm, as well as state-of-the-art methods, can handle larger data sets with the use of the properties than those currently possible without them.

Keywords: Bayesian networks, structure learning, properties of decomposable scores, structural constraints, branch-and-bound technique

1. Introduction

A Bayesian network is a probabilistic graphical model that relies on a structured dependency among random variables to represent a joint probability distribution in a compact and efficient manner. It is composed by a directed acyclic graph (DAG) where nodes are associated to random variables and conditional probability distributions are defined for variables given their parents in the graph. Learning the graph (or structure) of these networks from data is one of the most challenging problems, even if data are complete. The problem is known to be NP-hard (Chickering et al., 2004), and best exact known methods take exponential time on the number of variables and are applicable to small settings (around 30 variables). Approximate procedures can handle larger networks, but usually they get stuck in local maxima. Nevertheless, the quality of the structure plays a crucial role in the accuracy of the model. If the dependency among variables is not properly learned, the estimated distribution may be far from the *correct* one.

In general terms, the problem is to find the best structure (DAG) according to some score function that depends on the data (Heckerman et al., 1995). There are methods based on other (local) statistical analysis (Spirtes et al., 1993), but they follow a completely different approach. The research on this topic is active (Chickering, 2002; Teyssier and Koller, 2005; Tsamardinos et al., 2006; Silander and Myllymaki, 2006; Parviainen and Koivisto, 2009; de Campos et al., 2009; Jaakkola et al., 2010), mostly focused on complete data. In this case, best exact ideas (where it is guaranteed to find the global best scoring structure) are based on dynamic programming (Koivisto and Sood, 2004; Singh and Moore, 2005; Koivisto, 2006; Silander and Myllymaki, 2006; Parviainen and Koivisto, 2009), and they spend time and memory proportional to $n \cdot 2^n$, where n is the number of variables. Such complexity forbids the use of those methods to a couple of tens of variables, mainly because of the memory consumption (even though time complexity is also a clear issue). Ott and Miyano (2003) devise a faster algorithm when the complexity of the structure is limited (for instance the maximum number of parents per node and the degree of connectivity of a subjacent graph). Perrier et al. (2008) use structural constraints (creating a super-structure from which the optimal must be a subgraph) to reduce the search space, showing that such direction is promising when one wants to learn structures of large data sets. Kojima et al. (2010) extend the same ideas with other types of constraints. Mostly these methods are based on improving the dynamic programming method to work over reduced search spaces. On a different front, Jaakkola et al. (2010) apply a linear programming relaxation to solve the problem, together with a branch-and-bound search. Branch-and-bound methods can be effective when good bounds and cuts are available. For example, this has happened with certain success in the Traveling Salesman Problem (Applegate et al., 2006). We have proposed an algorithm that also uses branch and bound, but employs a different technique to find bounds (de Campos et al., 2009). It has been showed that branch and bound methods can handle somewhat larger networks than the dynamic programming ideas. The method is described in detail in Section 5.

In the first part of this paper, we present structural constraints as a way to reduce the search space. We explore the use of constraints to devise methods to learn specialized versions of Bayesian networks (such as naive Bayes and Tree-augmented naive Bayes) and generalized versions, such as Dynamic Bayesian networks (DBNs). DBNs are used to model temporal processes. We describe a procedure to map the structural learning problem of a DBN into a corresponding augmented Bayesian network through the use of further constraints, so that the same exact algorithm we discuss for Bayesian networks can be employed for DBNs.

In the second part, we present some properties of the problem that bring a considerable improvement on many known methods. We build on our recent work (de Campos et al., 2009) on *Akaike Information Criterion* (AIC) and *Bayesian Information Criterion* (BIC), and present new results for the Bayesian Dirichlet (BD) criterion (Cooper and Herskovits, 1992) and some derivations under a few assumptions. We show that the search space of possible structures can be reduced drastically without losing the global optimality guarantee and that the memory requirements are very small in many practical cases.

As data sets with many variables cannot be efficiently handled (unless P=NP), a desired property of a learning method is to produce an *any-time* solution, that is, the procedure, if stopped at any moment, provides an approximate solution, while if run until it finishes, a global optimum solution is found. However, the most efficient exact methods are not *any-time*. We describe an any-time exact algorithm using a branch-and-bound (B&B) approach with caches. Scores are pre-computed during an initialization step to save computational time. Then we perform the search over the possible graphs iterating over arcs. Because of the B&B properties, the algorithm can be stopped with a best current solution and an upper bound to the global optimum, which gives a certificate to the answer and allows the user to stop the computation when she/he believes that the current solution is good enough. For example, such an algorithm can be integrated with a structural Expectation–Maximization (EM) method without the huge computational expenses of other exact methods by using the generalized EM (where finding an improving solution is enough), but still guaranteeing that a global optimum is found if run until the end. Due to this property, the only source of approximation would regard the EM method itself. It worth noting that using a B&B method is not new for structure learning (Suzuki, 1996). Still, that previous idea does not constitute a global exact algorithm, instead the search is conducted after a node ordering is fixed. Our method does not rely on a predefined ordering and finds a global optimum structure considering all possible orderings.

The paper is divided as follows. Section 2 describes the notation and introduces Bayesian networks and the structure learning problem based on score functions. Section 3 presents the structural constraints that are treated in this work, and shows examples on how they can be used to learn different types of networks. Section 4 presents important properties of the score functions that considerably reduce the memory and time costs of many methods. Section 5 details our branch-and-bound algorithm, while Section 6 shows experimental evaluations of the properties, the constraints and the exact method. Finally, Section 7 concludes the paper.

2. Bayesian networks

A Bayesian network represents a joint probability distribution over a collection of random variables, which we assume to be categorical. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} \doteq (V_\mathcal{G}, E_\mathcal{G})$ is a directed acyclic graph (DAG) with $V_\mathcal{G}$ a collection of n nodes associated to random variables $\mathcal{X}$ (a node per variable), and $E_\mathcal{G}$ a collection of arcs; $\mathcal{P}$ is a collection of conditional mass functions $p(X_i|\Pi_i)$ (one for each instantiation of Π_i), where Π_i denotes the parents of X_i in the graph (Π_i may be empty), respecting the relations of $E_\mathcal{G}$. In a Bayesian network every variable is conditionally independent of its non-descendants given its parents (Markov condition).

We use uppercase letters such as X_i, X_j to represent variables (or nodes of the graph, which are used interchanged), and x_i to represent a generic state of X_i, which has state space $\Omega_{X_i} \doteq \{x_{i1}, x_{i2}, \ldots, x_{ir_i}\}$, where $r_i \doteq |\Omega_{X_i}| \geq 2$ is the number of (finite) categories of X_i ($|\cdot|$ is the cardinality of a set or vector, and the notation $\doteq$ is used to indicate a definition instead of a mathematical equality). Bold letters are used to emphasize sets or vectors. For example, $\mathbf{x} \in \Omega_\mathbf{X} \doteq \times_{X \in \mathbf{X}} \Omega_X$, for $\mathbf{X} \subseteq \mathcal{X}$, is an instantiation for all the variables in $\mathbf{X}$. Furthermore, $r_{\Pi_i} \doteq |\Omega_{\Pi_i}| = \prod_{X_t \in \Pi_i} r_t$ is the number of possible instantiations of the parent set Π_i of X_i, and $\boldsymbol{\theta} = (\theta_{ijk})_{\forall ijk}$ is the entire vector of parameters such that the elements are $\theta_{ijk} = p(x_{ik}|\boldsymbol{\pi}_{ij})$, with $i \in \{1, \ldots, n\}$, $j \in \{1, ..., r_{\Pi_i}\}$, $k \in \{1, ..., r_i\}$, and $\boldsymbol{\pi}_{ij} \in \Omega_{\Pi_i}$.

Because of the Markov condition, the Bayesian network represents a joint probability distribution by the expression $p(\mathbf{x}) = p(x_1, \ldots, x_n) = \prod_i p(x_i|\boldsymbol{\pi}_i)$, for every $\mathbf{x} \in \Omega_{\mathcal{X}}$, where every x_i and $\boldsymbol{\pi}_i$ are consistent with $\mathbf{x}$.

Given a complete data set $D = \{D_1, \ldots, D_N\}$ with N instances, where $D_u \doteq \mathbf{x}_u \in \Omega_{\mathcal{X}}$ is an instantiation of all the variables, the goal of structure learning is to find a DAG $\mathcal{G}$ that maximizes a given score function, that is, we look for $\mathcal{G}^* = \text{argmax}_{\mathcal{G} \in \boldsymbol{\mathcal{G}}}\, s_D(\mathcal{G})$, with $\boldsymbol{\mathcal{G}}$ the set of all DAGs with nodes $\mathcal{X}$, for a given score function s_D (the dependency on data is indicated by the subscript D).[1] In this paper, we consider some well-known score functions: the Bayesian Information Criterion (BIC) (Schwarz, 1978) (which is equivalent to the *Minimum Description Length*), the Akaike Information Criterion (AIC) (Akaike, 1974), and the Bayesian Dirichlet (BD) (Cooper and Herskovits, 1992), which has as subcases BDe and BDeu (Buntine, 1991; Cooper and Herskovits, 1992; Heckerman et al., 1995). As done before in the literature, we assume parameter independence and modularity (Heckerman et al., 1995). The score functions based on BIC and AIC differ only in the weight that is given to the penalty term:

$$BIC/AIC: \quad s_D(\mathcal{G}) = \max_{\boldsymbol{\theta}} L_{\mathcal{G},D}(\boldsymbol{\theta}) - t(\mathcal{G}) \cdot w,$$

where $t(\mathcal{G}) = \sum_{i=1}^{n}(r_{\Pi_i} \cdot (r_i - 1))$ is the number of free parameters, $w = \frac{\log N}{2}$ for BIC and $w = 1$ for AIC, $L_{\mathcal{G},D}$ is the log-likelihood function with respect to data D and graph $\mathcal{G}$:

$$L_{\mathcal{G},D}(\boldsymbol{\theta}) = \log \prod_{i=1}^{n} \prod_{j=1}^{r_{\Pi_i}} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}}, \tag{1}$$

where n_{ijk} indicates how many elements of D contain both x_{ik} and $\boldsymbol{\pi}_{ij}$. Note that the values $(n_{ijk})_{\forall ijk}$ depend on the graph $\mathcal{G}$ (more specifically, they depend on the parent set Π_i of each X_i), so a more precise notation would be to use $n_{ijk}^{\Pi_i}$ instead of n_{ijk}. We avoid this heavy notation for simplicity unless necessary in the context. Moreover, we know that $\boldsymbol{\theta}^* = (\theta^*_{ijk})_{\forall ijk} = (\frac{n_{ijk}}{n_{ij}})_{\forall ijk} = \text{argmax}_{\boldsymbol{\theta}}\, L_{\mathcal{G},D}(\boldsymbol{\theta})$, with $n_{ij} = \sum_k n_{ijk}$.[2]

In the case of the BD criterion, the idea is to compute a score based on the posterior probability of the structure $p(\mathcal{G}|D)$. For that purpose, the following score function is used:

$$BD: \quad s_D(\mathcal{G}) = \log \left(p(\mathcal{G}) \cdot \int p(D|\mathcal{G}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}|\mathcal{G}) d\boldsymbol{\theta} \right),$$

where the logarithmic is often used to simplify computations, $p(\boldsymbol{\theta}|\mathcal{G})$ is the prior of $\boldsymbol{\theta}$ for a given graph $\mathcal{G}$, assumed to be a Dirichlet with hyper-parameters $\boldsymbol{\alpha} = (\alpha_{ijk})_{\forall ijk}$ (which are assumed to be strictly positive):

$$p(\boldsymbol{\theta}|\mathcal{G}) = \prod_{i=1}^{n} \prod_{j=1}^{r_{\Pi_i}} \Gamma(\alpha_{ij}) \prod_{k=1}^{r_i} \frac{\theta_{ijk}^{\alpha_{ijk}-1}}{\Gamma(\alpha_{ijk})},$$

1. In case of many optimal DAGs, then we assume to have no preference and argmax returns one of them.
2. If $n_{ij} = 0$, then $n_{ijk} = 0$ and we assume the fraction $\frac{n_{ijk}}{n_{ij}}$ to be equal to one.

where $\alpha_{ij} = \sum_k \alpha_{ijk}$. Hyper-parameters $(\alpha_{ijk})_{\forall ijk}$ also depend on the graph $\mathcal{G}$, and we indicate it by $\alpha_{ijk}^{\Pi_i}$ if necessary in the context. From now on, we also omit the subscript D. We assume that there is no preference for any graph, so $p(\mathcal{G})$ is uniform and vanishes in the computations. Under the assumptions, it has been shown (Cooper and Herskovits, 1992) that for multinomial distributions,

$$s(\mathcal{G}) = \log \prod_{i=1}^{n} \prod_{j=1}^{r_{\Pi_i}} \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})}. \tag{2}$$

The BDe score (Heckerman et al., 1995) assumes that $\alpha_{ijk} = \alpha^* \cdot p(\theta_{ijk}|\mathcal{G})$, where α^* is the hyper-parameter known as the Equivalent Sample Size (ESS), and $p(\theta_{ijk}|\mathcal{G})$ is the prior probability for $(x_{ik} \wedge \boldsymbol{\pi}_{ij})$ given $\mathcal{G}$ (or simply given Π_i). The BDeu score (Buntine, 1991; Cooper and Herskovits, 1992) assumes further that local priors are such that α_{ijk} becomes $\frac{\alpha^*}{r_{\Pi_i} r_i}$ and α^* is the only free hyper-parameter.

An important property of all such criteria is that their functions are decomposable and can be written in terms of the local nodes of the graph, that is, $s(\mathcal{G}) = \sum_{i=1}^{n} s_i(\Pi_i)$, such that

$$BIC/AIC: \quad s_i(\Pi_i) = \max_{\boldsymbol{\theta}_i} L_{\Pi_i}(\boldsymbol{\theta}_i) - t_i(\Pi_i) \cdot w, \tag{3}$$

where $L_{\Pi_i}(\boldsymbol{\theta}_i) = \sum_{j=1}^{r_{\Pi_i}} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk}$, and $t_i(\Pi_i) = r_{\Pi_i} \cdot (r_i - 1)$. And similarly,

$$BD: \quad s_i(\Pi_i) = \sum_{j=1}^{r_{\Pi_i}} \left(\log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} + \sum_{k=1}^{r_i} \log \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \right). \tag{4}$$

In the case of BIC and AIC, Equation (3) is used to compute the global score of a graph using the local scores at each node, while Equation (4) is employed for BD, BDe and BDeu, using the respective hyper-parameters $\boldsymbol{\alpha}$.

3. Structural Constraints

A way to reduce the space of possible DAGs is to consider some constraints provided by experts. We work with structural constraints that specify where arcs may or may not be included. These constraints help to reduce the search space and are available in many situations. Moreover, we show examples in Sections 3.1 and 3.2 of how these constraints can be used to learn structures of different types of networks, such as naive Bayes, tree-augmented naive Bayes, and Dynamic Bayesian networks. We work with the following rules, used to build up the structural constraints:

- *indegree*(X_j, k, op), where $op \in \{\text{lt}, \text{eq}\}$ and k an integer, means that the node X_j must have *less than* (when $op = \text{lt}$) or *equal to* (when $op = \text{eq}$) k parents.
- *arc*(X_i, X_j) indicates that the node X_i must be a parent of X_j.
- Operators *or* $(\vee)$ and *not* $(\neg)$ are used to form the rules. The *and* operator is not explicitly used as we assume that each constraint is in disjunctive normal form.

The structural constraints can be imposed locally as long as they involve just a single node and its parents. In essence, parent sets of a node X_i that do violate a constraint are never processed nor stored, and this can be checked locally when one is about to compute the local score. On the other hand, constraints such as $(arc(X_1, X_2) \vee arc(X_2, X_3))$ cannot be imposed locally, as it defines a non-local condition (the arcs go to distinct variables, namely X_2 and X_3). In this work we assume that constraints are local. Besides constraints devised by an expert, one might use constraints to force the learning procedure to obtain specialized types of networks. The next two subsections describe (somewhat non-trivial) examples of use of constraints to learn different types of networks. Specialized networks tend to be easier to learn, because the search space is already reduced to the structures that satisfy the underlying constraints. Notwithstanding, the readers who are only interested in learning general Bayesian networks might want to skip the rest of this section and continue from Section 4.

3.1 Learning Naive and TAN structures

For example, the constraints $\forall_{i\neq c, j\neq c}\ \neg arc(X_i, X_j)$ and $indegree(X_c, 0, \mathrm{eq})$ impose that only arcs from node X_c to the others are possible, and that X_c is a root node, that is, a Naive Bayes structure will be learned. A learning procedure would in fact act as a feature selection procedure by letting some variables unlinked. Note that the symbol $\forall$ just employed is not part of the language but is used for easy of expose (in fact it is necessary to write down every constraint defined by such construction). As another example, the constraints $\forall_{j\neq c}\ indegree(X_j, 3, \mathrm{lt})$, $indegree(X_c, 0, \mathrm{eq})$, and $\forall_{j\neq c}\ indegree(X_j, 0, \mathrm{eq}) \vee arc(X_c, X_j)$ ensure that all nodes have X_c as parent, or no parent at all. Besides X_c, each node may have at most one other parent, and X_c is a root node. This learns the structure of a Tree-augmented Naive (TAN) classifier, also performing a kind of feature selection (some variables may end up unlinked). In fact, it learns a forest of trees, as we have not imposed that all variables must be linked. In Section 6 we present some experimental results which indicate that learning TANs is a much easier (still very important) practical situation.

We point out that learning structures of networks with the particular purpose of building a classifier can be also tackled by other score functions that consider conditional distributions (Pernkopf and Bilmes, 2005). Here we present a way to learn TANs considering the fit of the joint distribution, which can be done by constraints. Further discussions about learning classifiers is not the aim of this work.

3.2 Learning Dynamic Bayesian Networks

A more sophisticated application of structural constraints is presented in this section, where they are employed to translate the structure learning in Dynamic Bayesian Networks (DBNs) to a corresponding problem in Bayesian networks. While Bayesian networks are not directly related to time, DBNs are used to model temporal processes. Assuming Markovian and stationary properties, DBNs may be encoded in a very compact way and inferences are executed quickly. They are built over a collection of sets of random variables $\{\mathcal{X}^0, \mathcal{X}^1, \ldots, \mathcal{X}^T\}$ representing variables in different times $0, 1, \ldots, T$ (we assume that time is discrete). A Markovian property holds, which ensures that $p(\mathcal{X}^{t+1}|\mathcal{X}^0, \ldots \mathcal{X}^t) = p(\mathcal{X}^{t+1}|\mathcal{X}^t)$, for $0 \leq t < T$. Furthermore, because the process is assumed to be stationary, we have that $p(\mathcal{X}^{t+1}|\mathcal{X}^t)$ is independent of t, that is, $p(\mathcal{X}^{t+1}|\mathcal{X}^t) = p(\mathcal{X}^{t'+1}|\mathcal{X}^{t'})$ for any $0 \leq t, t' < T$. This means that a DBN is just as a collection of Bayesian networks that share the same structure and parameters (apart from the initial Bayesian network for time zero). If $X_i^t \in \mathcal{X}^t$ are the variables at time t, a DBN may have arcs between nodes X_i^t of the same time t and arcs from nodes X_i^{t-1} (previous time) to nodes X_i^t of time t. Hence, a DBN can be viewed as two-slice temporal Bayesian network, where at time zero, we have a standard Bayesian network as in Section 2, which we denote $\mathcal{B}^o$, and for slices 1 to T we have another Bayesian network (called *transitional* Bayesian network and denoted simply $\mathcal{B}$) defined over the same variables but where nodes may have parents on two consecutive slices, that is, $\mathcal{B}$ precisely defines the distributions $p(\mathcal{X}^{t+1}|\mathcal{X}^t)$, for any $0 \leq t < T$.

To learn a DBN, we assume that many temporal sequences of data are available. Thus, a complete data set $D = \{D_1, \ldots, D_N\}$ is composed of N sequences, where each D_u is composed of instances $D_u^t \doteq \mathbf{x_u^t} = \{x_{u,1}^t, \ldots, x_{u,n}^t\}$, for $t = 0, \ldots, T$ (where T is the total number of slices/frames apart from the initial one). Note that there is an implicit order among the elements of each D_u. We denote by $D^0 \doteq \{D_u^0 : 1 \leq u \leq N\}$ the data of the first slice, and by $D^t \doteq \{(D_u^t, D_u^{t-1}) : 1 \leq u \leq N\}$, with $1 \leq t \leq T$, the data of a slice t (note that the data of the slice $t-1$ is also included, because it is necessary for learning the transitions). As the conditional probability distributions for time $t > 0$ share the same parameters, we can unroll the DBN to obtain the factorization $p(\mathcal{X}^{1:T}) = \prod_i p^0(X_i^0|\Pi_i^0) \prod_{t=1}^T \prod_i p(X_i^t|\Pi_i^t)$, where $p^0(X_i^0|\Pi_i^0)$ are the local conditional distributions of $\mathcal{B}^0$, X_i^t and Π_i^t represent the corresponding variables in time t, and $p(X_i^t|\Pi_i^t)$ are the local distributions of $\mathcal{B}$.

Unfortunately learning a DBN is at least as hard as learning a Bayesian network, because the former can be viewed as a generalization of the latter. Still, we show that the same method used for Bayesian networks can be used to learn DBNs. With complete data, learning parameters of DBNs is similar to learning parameters of Bayesian networks, but we deal with counts n_{ijk} for both $\mathcal{B}^0$ and $\mathcal{B}$. The counts related to $\mathcal{B}^0$ are obtained from the first slice of each sequence, so there are N samples overall, while counts for $\mathcal{B}$ are obtained from the whole time sequences, so there are $N \cdot T$ elements to consider (supposing that each sequence has the same length T, for ease of expose). The score function of a given structure decomposes between the score function of $\mathcal{B}^0$ and the score function of $\mathcal{B}$ (because of the decomposability of score functions), so we look for graphs such that

$$(\mathcal{G}^{0*}, \mathcal{G}'^*) = \underset{\mathcal{G}^0, \mathcal{G}'}{\operatorname{argmax}} \left(s_{D^0}(\mathcal{G}^0) + s_{D^{1:T}}(\mathcal{G}') \right) = (\underset{\mathcal{G}^0}{\operatorname{argmax}}\, s_{D^0}(\mathcal{G}^0), \underset{\mathcal{G}'}{\operatorname{argmax}}\, s_{D^{1:T}}(\mathcal{G}')), \tag{5}$$

where G^0 is a graph over $\mathcal{X}^0$ and $\mathcal{G}'$ is a graph over variables $\mathcal{X}^t, \mathcal{X}^{t-1}$ of a generic slice t and its predecessor $t-1$. Counts are obtained from data sets with time sequences separately for the initial and the transitional Bayesian networks, and the problem reduces to the learning problem in a Bayesian network with some constraints that force the arcs to respect the DBN's stationarity and Markovian characteristics (of course, it is necessary to obtain the counts from the data in a particular way). We make use of the constraints defined in Section 3 to develop a simple transformation of the structure learning problem to a corresponding structure learning problem in an augmented Bayesian network. The steps of this procedure are as follows:

1. Learn $\mathcal{B}^0$ using the data set D^0. Note that this is already a standard Bayesian network structure learning problem, so we obtain the graph $\mathcal{G}^0$ for the first maximization of Equation (5).

2. Suppose there is a Bayesian network $\mathcal{B}' = (\mathcal{G}', \mathcal{X}', \mathcal{P}')$ with twice as many nodes as $\mathcal{B}^0$. Denote the nodes as $(X_1, \ldots, X_n, X'_1, \ldots, X'_n)$. Construct a new data set D' that is composed by $N \cdot T$ elements $\{D^1, \ldots, D^T\}$. Note that D' is precisely a data set over $2n$ variables, because it is formed of pairs (D_u^{t-1}, D_u^t), which are complete instantiations for the variables of $\mathcal{B}'$, containing the elements of two consecutive slices.

3. Include structural constraints as follows:

$$\forall_{1\leq i\leq n}\ arc(X_i, X'_i) \tag{6}$$
$$\forall_{1\leq i\leq n}\ indegree(X_i, 0, eq) \tag{7}$$

   Equation (6) forces the time relation between the same variable in consecutive time slices (in fact this constraint might be discarded if someone does not want to enforce each variable to be correlated to itself of the past slice). Equation (7) forces the variables $X_1, \ldots, X_n$ to have no parents (these are the variables that are simulating the previous slice, while the variables $\mathcal{X}'$ are simulating the current slice).

4. Learn $\mathcal{B}'$ using the data set D' with an standard Bayesian network structure learning procedure, capable of enforcing the structural constraints. Note that the parent sets of $X_1, \ldots, X_n$ are already fixed to be empty, so the output graph will maximize the scores associated only to nodes $\mathcal{X}'$: $\text{argmax}_{\mathcal{G}'}\, s_{D^{1:T}}(\mathcal{G}')) =$

$$\operatorname*{argmax}_{\mathcal{G}'} \left(\sum_i s_{i,D^{1:T}}(\Pi_i) + \sum_{i'} s_{i',D^{1:T}}(\Pi'_i) \right) = \operatorname*{argmax}_{\mathcal{G}'} \sum_{i'} s_{i',D^{1:T}}(\Pi'_i).$$

   This holds because of the decomposability of the score function among nodes, so that the scores of the nodes $X_1, \ldots, X_n$ are fixed and can be disregarded in the maximization (they are constant).

5. Finally, we take the subgraph of $\mathcal{G}'$ corresponding to the variables $X'_1, \ldots, X'_n$ to be the graph of the transitional Bayesian network $\mathcal{B}$. This subgraph has arcs among $X'_1, \ldots, X'_n$ (which are arcs correlating variables of the same time slice) as well as arcs from the previous slice to the nodes $X'_1, \ldots, X'_n$.

Therefore, after applying this transformation, the structure learning problem in a DBN can be performed by two calls to the method that solves the problem in a Bayesian network. We point out that an expert may create her/his own constraints to be used during the learning, besides those constraints introduced by the transformation, as long as such constraints do not violate the DBN implicit constraints. This makes possible to learn DBNs together with expert's knowledge in the form of structural constraints.

4. Properties of the score functions

In this section we present mathematical properties that are useful when computing score functions. Local scores need to be computed many times to evaluate the candidate graphs when we look for the best graph. Because of decomposability, we can avoid to compute such functions several times by creating a cache that contains $s_i(\Pi_i)$ for each X_i and each parent set Π_i. Note that this cache may have an exponential size on n, as there are 2^{n-1} subsets of $\{X_1, \ldots, X_n\} \setminus \{X_i\}$ to be considered as parent sets. This gives a total space and time of $O(n \cdot 2^n \cdot v)$ to build the cache, where v is the worse case asymptotic time to compute the local score function at each node.[3] Instead, we describe a collection of results that are used to obtain much smaller caches in many practical cases.

First, Lemma 1 is quite simple but very useful to discard elements from the cache of each node X_i. It holds for all score functions that we treat in this paper. It was previously stated in Teyssier and Koller (2005) and de Campos et al. (2009), among others.

Lemma 1 *Let X_i be a node of $\mathcal{G}'$, a candidate DAG for a Bayesian network where the parent set of X_i is Π'_i. Suppose $\Pi_i \subset \Pi'_i$ is such that $s_i(\Pi_i) > s_i(\Pi'_i)$ (where s is one of BIC, AIC, BD or derived criteria). Then Π'_i is not the parent set of X_i in an optimal DAG $\mathcal{G}^*$.*

Proof This fact comes straightforward from the decomposability of the score functions. Take a graph $\mathcal{G}$ that differs from $\mathcal{G}'$ only on the parent set of X_i, where it has Π_i instead of Π'_i. Note that $\mathcal{G}$ is also a DAG (as $\mathcal{G}$ is a subgraph of $\mathcal{G}'$ built from the removal of some arcs, which cannot create cycles) and $s(\mathcal{G}) = \sum_{j \neq i} s_j(\Pi'_j) + s_i(\Pi_i) > \sum_{j \neq i} s_j(\Pi'_j) + s_i(\Pi'_i) = s(\mathcal{G}')$. Any DAG $\mathcal{G}'$ with parent set Π'_i for X_i has a subgraph $\mathcal{G}$ with a better score than that of $\mathcal{G}'$, and thus Π'_i is not the optimal parent configuration for X_i in $\mathcal{G}^*$. ■

Unfortunately Lemma 1 does not tell us anything about supersets of Π'_i, that is, we still need to compute scores for all the possible parent sets and later verify which of them can be removed. This would still leave us with $n \cdot 2^n \cdot v$ asymptotic time and space requirements (although the space would be reduced after applying the lemma). The next two subsections present results to avoid all such computations. BIC and AIC are treated separately from BD and derivatives (reasons for that will become clear in the derivations).

4.1 BIC and AIC score properties

Next theorems handle the issue of having to compute scores for all possible parent sets, when one is using BIC or AIC criteria. BD scores are dealt later on.

Theorem 2 *Using BIC or AIC as score function, suppose that X_i, Π_i are such that $r_{\Pi_i} > \frac{N}{w} \frac{\log r_i}{r_i - 1}$. If Π'_i is a proper superset of Π_i, then Π'_i is not the parent set of X_i in an optimal structure.*

3. Note that the time to compute a single local score might be large depending on the number of parents but still asymptotically bounded by the data set size.

Proof [4] We know that Π'_i contains at least one additional node, that is, $\Pi'_i \supseteq \Pi_i \cup \{X_e\}$ and $X_e \notin \Pi_i$. Because $\Pi_i \subset \Pi'_i$, $L_i(\Pi'_i)$ is certainly greater than or equal to $L_i(\Pi_i)$, and $t_i(\Pi'_i)$ will certainly be greater than the corresponding value $t_i(\Pi_i)$ in $\mathcal{G}$. The difference in the scores is $s_i(\Pi'_i) - s_i(\Pi_i)$, which equals to (see the explanations after the formulas):

$$\begin{aligned}
\max_{\boldsymbol{\theta}'_i} L_i(\Pi'_i) - t_i(\Pi'_i) - (\max_{\boldsymbol{\theta}_i} L_i(\Pi_i) - t_i(\Pi_i)) \leq \\
-\max_{\boldsymbol{\theta}_i} L_i(\Pi_i) - t_i(\Pi'_i) + t_i(\Pi_i) = \\
\sum_{j=1}^{r_{\Pi_i}} n_{ij} \left(-\sum_{i=1}^{r_i} \frac{n_{ijk}}{n_{ij}} \log \frac{n_{ijk}}{n_{ij}} \right) - t_i(\Pi'_i) + t_i(\Pi_i) \leq \\
\sum_{j=1}^{r_{\Pi_i}} n_{ij} H(\theta_{ij}) - t_i(\Pi'_i) + t_i(\Pi_i) \leq \\
\sum_{j=1}^{r_{\Pi_i}} n_{ij} \log r_i - r_{\Pi_i} \cdot (r_e - 1) \cdot (r_i - 1) \cdot w \leq \\
\sum_{j=1}^{r_{\Pi_i}} n_{ij} \log r_i - r_{\Pi_i} \cdot (r_i - 1) \cdot w = N \log r_i - r_{\Pi_i} \cdot (r_i - 1) \cdot w.
\end{aligned}$$

The first step uses the fact that $L_i(\Pi'_i)$ is negative, so we drop it, the second step uses the fact that $\theta^*_{ijk} = \frac{n_{ijk}}{n_{ij}}$, with $n_{ij} = \sum_{i=1}^{r_i} n_{ijk}$, the third step uses the definition of entropy $H(\cdot)$ of a discrete distribution, and the fourth step uses the fact that the entropy of a discrete distribution is less than the log of its number of categories. Finally, the last equation is negative if $r_{\Pi_i} \cdot (r_i - 1) \cdot w > N \log r_i$, which is exactly the hypothesis of the theorem. Hence $s_i(\Pi'_i) < s_i(\Pi_i)$, and Lemma 1 guarantees that Π'_i cannot be the parent set of X_i in an optimal structure. ■

Corollary 3 *Using BIC or AIC as criterion, the optimal graph $\mathcal{G}$ has at most $O(\log N)$ parents per node.*

Proof Assuming $N > 4$, we have $\frac{\log r_i}{w(r_i - 1)} < 1$ (because w is either 1 or $\frac{\log N}{2}$). Take a variable X_i and a parent set Π_i with exactly $\lceil \log_2 N \rceil$ elements. Because every variable has at least two states, we know that $r_{\Pi_i} \geq 2^{|\Pi_i|} \geq N > \frac{N}{w} \frac{\log r_i}{r_i - 1}$, and by Theorem 2 we know that no proper superset of Π_i can be an optimal parent set. ■

Theorem 2 and Corollary 3 ensures that the cache stores at most $O(\binom{n-1}{\log N})$ elements for each variable (all combinations up to $\lceil \log_2 N \rceil$ parents). Next lemma does not help us to improve the theoretical size bound that is achieved by Corollary 3, but it is quite useful in practice because it is applicable even in cases where Theorem 2 is not, implying that less number of parent sets need to be inspected.

4. Another similar proof appears in Bouckaert (1994), but it leads directly to the conclusion of Corollary 3. The intermediate result is algorithmically important.

Theorem 4 *Let BIC or AIC be the score criterion and let X_i be a node with $\Pi_i \subset \Pi'_i$ two possible parent sets such that $t_i(\Pi'_i) + s_i(\Pi_i) > 0$. Then Π'_i and all supersets $\Pi''_i \supset \Pi'_i$ are not optimal parent configurations for X_i.*

Proof We have that $t_i(\Pi'_i) + s_i(\Pi_i) > 0 \Rightarrow -t_i(\Pi'_i) - s_i(\Pi_i) < 0$, and because $L_i(\cdot)$ is a negative function, it implies

$$\Rightarrow (L_i(\Pi'_i) - t_i(\Pi'_i)) - s_i(\Pi_i) < 0 \Rightarrow s_i(\Pi'_i) < s_i(\Pi_i).$$

Using Lemma 1, we have that Π'_i is not the optimal parent set for X_i. The result also follows for any $\Pi''_i \supset \Pi_i$, as we know that $t_i(\Pi''_i) > t_i(\Pi'_i)$ and the same argument suffices. ■

Theorem 4 provides a bound to discard parent sets without even inspecting them. The idea is to verify the assumptions of Theorem 4 every time the score of a parent set Π_i of X_i is about to be computed by taking the best score of any subset and testing it against the theorem. Only subsets that have been checked against the structural constraints can be used, that is, a subset with high score but that violates constraints cannot be used as the "certificate" to discard its supersets (in fact, it is not a valid parent set at first). This ensures that the results are valid even in the presence of constraints. Whenever the theorem can be applied, Π_i is discard and all its supersets are not even inspected. This result allows us to stop computing scores earlier than the worst-case, reducing the number of computations to build and store the cache. Π_i is also checked against Lemma 1 (which is stronger in the sense that instead of a bounding function, the actual scores are directly compared). However Lemma 1 cannot help us to avoid analyzing the supersets of Π_i.

4.2 BD score properties

First note that the BD scores can be rewritten as:

$$s_i(\Pi_i) = \sum_{j \in J_i} \left(\log \frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij} + n_{ij})} + \sum_{k \in K_{ij}} \log \frac{\Gamma(\alpha_{ijk} + n_{ijk})}{\Gamma(\alpha_{ijk})} \right),$$

where $J_i \doteq J_i^{\Pi_i} \doteq \{1 \leq j \leq r_{\Pi_i} : \; n_{ij} \neq 0\}$, because $n_{ij} = 0$ implies that all terms cancel each other. In the same manner, $n_{ijk} = 0$ implies that the terms of the internal summation cancel out, so let $K_{ij} \doteq K_{ij}^{\Pi_i} \doteq \{1 \leq k \leq r_i : \; n_{ijk} \neq 0\}$ be the indices of the categories of X_i such that $n_{ijk} \neq 0$. Let $K_i^{\Pi_i} \doteq \cup_j K_{ij}^{\Pi_i}$ be a vector with all indices corresponding to non-zero counts for Π_i (note that the symbol $\cup$ must be seen as a concatenation of vectors, as we allow $K_i^{\Pi_i}$ to have repetitions). The counts n_{ijk} (and consequently $n_{ij} = \sum_k n_{ijk}$) are completely defined if we know the parent set Π_i. Rewrite the score as follows:

$$s_i(\Pi_i) = \sum_{j \in J_i} \left(f(K_{ij}, (\alpha_{ijk})_{\forall k}) + g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \right),$$

with

$$f(K_{ij},(\alpha_{ijk})_{\forall k}) = \log\Gamma(\alpha_{ij}) - \sum_{k\in K_{ij}} \log\Gamma(\alpha_{ijk}),$$

$$g((n_{ijk})_{\forall k},(\alpha_{ijk})_{\forall k}) = -\log\Gamma(\alpha_{ij}+n_{ij}) + \sum_{k\in K_{ij}} \log\Gamma(\alpha_{ijk}+n_{ijk}).$$

We do not need K_{ij} as argument of $g(\cdot)$ because the set of non-zero n_{ijk} is known from the counts $(n_{ijk})_{\forall k}$ that are already available as arguments of $g(\cdot)$. To achieve the desired theorem that will be able to reduce the computational time to build the cache, some intermediate results are necessary.

Lemma 5 *Let Π_i be the parent set of X_i, $(\alpha_{ijk})_{\forall ijk} > 0$ be the hyper-parameters, and integers $(n_{ijk})_{\forall ijk} \geq 0$ be counts obtained from data. We have that $g((n_{ijk})_{\forall k},(\alpha_{ijk})_{\forall k}) \leq -\log\Gamma(v) \approx 0.1214$ if $n_{ij} \geq 1$, where $v = \operatorname{argmax}_{x>0} -\log\Gamma(x) \approx 1.4616$. Furthermore, $g((n_{ijk})_{\forall k},(\alpha_{ijk})_{\forall k}) \leq -\log\alpha_{ij} + \log\alpha_{ijk} - f(K_{ij},(\alpha_{ijk})_{\forall k})$ if $|K_{ij}| = 1$.*

Proof We use the relation $\Gamma(x+\sum_k a_k) \geq \Gamma(x+1)\prod_k \Gamma(a_k)$, for $x \geq 0$, $\forall_k a_k \geq 1$ and $\sum_k a_k \geq 1$ (note that it is valid even if there is a single element in the summation). This relation comes from the Beta function inequality:

$$\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \leq \frac{x+y}{xy} \implies \Gamma(x+1)\Gamma(y+1) \leq \Gamma(x+y+1),$$

where $x, y > 0$. Applying the transformation $y+1 = \sum_t a_t$ (which is possible because $\sum_t a_t > 1$ and thus $y > 0$), we obtain:

$$\Gamma(x+\sum_t a_t) \geq \Gamma(x+1)\Gamma(\sum_t a_t) \geq \Gamma(x+1)\prod_t \Gamma(a_t), \tag{8}$$

(the last step is due to $a_t \geq 1$ for all t, so the same relation of the Beta function can be overall applied, because $\Gamma(x+1)\Gamma(y+1) \leq \Gamma(x+y+1) \leq \Gamma(x+1+y+1)$).

With the relation just devised in hands, we have

$$\frac{\Gamma(\alpha_{ij}+n_{ij})}{\prod_{k\in K_{ij}}\Gamma(\alpha_{ijk}+n_{ijk})} = \frac{\Gamma(\sum_{1\leq k\leq r_i}(\alpha_{ijk}+n_{ijk}))}{\prod_{k\in K_{ij}}\Gamma(\alpha_{ijk}+n_{ijk})} =$$

$$= \frac{\Gamma(\sum_{k\notin K_{ij}}\alpha_{ijk} + \sum_{k\in K_{ij}}(\alpha_{ijk}+n_{ijk}))}{\prod_{k\in K_{ij}}\Gamma(\alpha_{ijk}+n_{ijk})} \geq \Gamma(1+\sum_{k\notin K_{ij}}\alpha_{ijk}),$$

obtained by renaming $x = \sum_{k\notin K_{ij}}\alpha_{ijk}$ and $a_k = \alpha_{ijk}+n_{ijk}$ (we have that $\sum_{k\in K_{ij}}(\alpha_{ijk}+n_{ijk}) \geq n_{ij} \geq 1$ and each $a_k \geq 1$). Thus

$$g((n_{ijk})_{\forall k},(\alpha_{ijk})_{\forall k}) = -\log\frac{\Gamma(\alpha_{ij}+n_{ij})}{\prod_{k\in K_{ij}}\Gamma(\alpha_{ijk}+n_{ijk})} \leq -\log\Gamma(1+\sum_{k\notin K_{ij}}\alpha_{ijk}).$$

Because $v = \operatorname{argmax}_{x>0} -\log\Gamma(x)$, we have $-\log\Gamma(1+\sum_{k\notin K_{ij}}\alpha_{ijk}) \leq -\log\Gamma(v)$.

Now, the second part of the lemma. If $|K_{ij}| = 1$, then let $K_{ij} = \{k\}$. We know that $n_{ij} \geq 1$ and thus

$$g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) = -\log \frac{\Gamma(\alpha_{ij} + n_{ij})}{\Gamma(\alpha_{ijk} + n_{ij})} = -\log \left(\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ijk})} \prod_{t=0}^{n_{ij}-1} \frac{(\alpha_{ij} + t)}{(\alpha_{ijk} + t)} \right) =$$

$$= -f(K_{ij}, (\alpha_{ijk})_{\forall k}) - \log \frac{\alpha_{ij}}{\alpha_{ijk}} - \sum_{t=1}^{n_{ij}-1} \log \frac{(\alpha_{ij} + t)}{(\alpha_{ijk} + t)} \leq -\log \alpha_{ij} + \log \alpha_{ijk} - f(K_{ij}, (\alpha_{ijk})_{\forall k}),$$

because $\frac{(\alpha_{ij}+t)}{(\alpha_{ijk}+t)} \geq 1$ for every t. ■

Lemma 6 *Let Π_i be the parent set of X_i, $(\alpha_{ijk})_{\forall ijk} > 0$ be the hyper-parameters, and integers $(n_{ijk})_{\forall ijk} \geq 0$ be counts obtained from data. We have that $g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) \leq 0$ if $n_{ij} \geq 2$.*

Proof If $n_{ij} \geq 2$, we use the relation $\Gamma(x + \sum_k a_k) \geq \Gamma(x+2) \prod_k \Gamma(a_k)$, for $x \geq 0$, $\forall_k a_k \geq 1$ and $\sum_k a_k \geq 2$. This inequality is obtained in the same way as in Lemma 5, but using a tighter Beta function bound:

$$\mathcal{B}(x, y) \leq \frac{x+y}{xy} \left(\frac{(x+1)(y+1)}{x+y+1} \right)^{-1} \implies \Gamma(x+2)\Gamma(y+2) \leq \Gamma(x+y+2),$$

and the relation follows by using $y + 2 = \sum_t a_t$ and the same derivation as before. Now,

$$\frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} = \frac{\Gamma(\sum_{1 \leq k \leq r_i} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} =$$

$$= \frac{\Gamma(\sum_{k \notin K_{ij}} \alpha_{ijk} + \sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}))}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \geq \Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk}),$$

obtained by renaming $x = \sum_{k \notin K_{ij}} \alpha_{ijk}$ and $a_k = \alpha_{ijk} + n_{ijk}$, as we know that $\sum_{k \in K_{ij}} (\alpha_{ijk} + n_{ijk}) \geq n_{ij} \geq 2$ and each $a_k \geq 1$. Finally,

$$g((n_{ijk})_{\forall k}, (\alpha_{ijk})_{\forall k}) = -\log \frac{\Gamma(\alpha_{ij} + n_{ij})}{\prod_{k \in K_{ij}} \Gamma(\alpha_{ijk} + n_{ijk})} \leq -\log \Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk}) \leq 0,$$

because $\Gamma(2 + \sum_{k \notin K_{ij}} \alpha_{ijk}) \geq 1$. ■

Lemma 7 *Given a BD score and two parent sets Π_i^0 and Π_i for a node X_i such that $\Pi_i^0 \subset \Pi_i$, if*

$$s_i(\Pi_i^0) > \sum_{\substack{j \in J_i^{\Pi_i}: \\ |K_{ij}^{\Pi_i}| \geq 2}} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{\substack{j \in J_i^{\Pi_i}: \\ |K_{ij}^{\Pi_i}| = 1}} \log \frac{\alpha_{ijk'}^{\Pi_i}}{\alpha_{ij}^{\Pi_i}}, \tag{9}$$

then Π_i is not the optimal parent set for X_i.

Proof Using the results of Lemmas 5 and 6,

$$\begin{aligned}
s_i(\Pi_i) &= \sum_{j \in J_i} \Big(f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + g((n_{ijk}^{\Pi_i})_{\forall k}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) \Big) \\
&\leq \sum_{j \in J_i:\ |K_{ij}^{\Pi_i}| \geq 2} \Big(f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + g((n_{ijk}^{\Pi_i})_{\forall k}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) \Big) + \\
&\qquad + \sum_{j \in J_i^{\Pi_i}:\ |K_{ij}^{\Pi_i}| = 1} \Big(-\log \alpha_{ij}^{\Pi_i} + \log \alpha_{ijk'}^{\Pi_i} \Big) \\
&\leq \sum_{j \in J_i^{\Pi_i}: |K_{ij}^{\Pi_i}| \geq 2} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{j \in J_i^{\Pi_i}: |K_{ij}^{\Pi_i}| = 1} \log \frac{\alpha_{ijk'}^{\Pi_i}}{\alpha_{ij}^{\Pi_i}},
\end{aligned}$$

which by the assumption of this lemma, is less than $s_i(\Pi_i^0)$. Thus, we conclude that the parent set Π_i^0 has better score than Π_i, and the desired result follows from Lemma 1. ■

Lemma 8 *Given the BDeu score, $(\alpha_{ijk})_{\forall ijk} > 0$, and integers $(n_{ijk})_{\forall ijk} \geq 0$ such that $\alpha_{ij} \leq 0.8349$ and $|K_{ij}| \geq 2$ for a given j, then $f(K_{ij}, (\alpha_{ijk})_{\forall k}) \leq -|K_{ij}| \cdot \log r_i$.*

Proof Using $\alpha_{ijk} \leq \alpha_{ij} \leq 0.8349$ (for all k), we have

$$\begin{aligned}
f(K_{ij}, (\alpha_{ijk})_{\forall k}) &= \log \Gamma(\alpha_{ij}) - |K_{ij}| \log \Gamma(\frac{\alpha_{ij}}{r_i}) \\
&= \log \Gamma(\alpha_{ij}) - |K_{ij}| \log \Gamma(\frac{\alpha_{ij}}{r_i} + 1) + |K_{ij}| \log \frac{\alpha_{ij}}{r_i} \\
&= \log \Gamma(\alpha_{ij}) - |K_{ij}| \log \frac{\Gamma(\frac{\alpha_{ij}}{r_i} + 1)}{\alpha_{ij}} - |K_{ij}| \log r_i \\
&= |K_{ij}| \log \frac{\Gamma(\alpha_{ij})^{1/|K_{ij}|} \alpha_{ij}}{\Gamma(\frac{\alpha_{ij}}{r_i} + 1)} - |K_{ij}| \log r_i.
\end{aligned}$$

Now, $\Gamma(\alpha_{ij})^{1/|K_{ij}|}\alpha_{ij} \leq \Gamma(\frac{\alpha_{ij}}{r_i} + 1)$, because $r_i \geq 2$, $|K_{ij}| \geq 2$ and $\alpha_{ij} \leq 0.8349$ (this number can be computed by numerically solving the inequality for $r_i = |K_{ij}| = 2$). We point out that 0.8349 is a bound for α_{ij} that ensures this last inequality to hold when $r_i = |K_{ij}| = 2$, which is the worst case scenario (greater values of r_i and $|K_{ij}|$ make the left-hand side decrease and the right-hand side increase). Because r_i of each node is known, tighter bounds might be possible according to the node. ■

Theorem 9 *Given the BDeu score and two parent sets Π_i^0 and Π_i for a node X_i such that $\Pi_i^0 \subset \Pi_i$ and $\alpha_{ij}^{\Pi_i} \leq 0.8349$ for every j, if $s_i(\Pi_i^0) > -|K_i^{\Pi_i}| \log r_i$ then neither Π_i nor any superset $\Pi_i' \supset \Pi_i$ are optimal parent sets for X_i.*

Proof We have that

$$s_i(\Pi_i^0) > -|K_i^{\Pi_i}| \log r_i = \sum_{j \in J_i^{\Pi_i}:\ |K_{ij}^{\Pi_i}| \geq 2} -|K_{ij}^{\Pi_i}| \log r_i + \sum_{j \in J_i^{\Pi_i}:\ |K_{ij}^{\Pi_i}| = 1} -\log r_i,$$

which by Lemma 8 is greater than or equal to

$$\sum_{j \in J_i^{\Pi_i}:\ |K_{ij}^{\Pi_i}| \geq 2} f(K_{ij}^{\Pi_i}, (\alpha_{ijk}^{\Pi_i})_{\forall k}) + \sum_{j \in J_i^{\Pi_i}:\ |K_{ij}^{\Pi_i}| = 1} -\log r_i.$$

Now, Lemma 7 suffices to show that Π_i is not a optimal parent set, because $-\log r_i = \log \frac{\alpha_{ijk}^{\Pi_i}}{\alpha_{ij}^{\Pi_i}}$ for any k. To show the result for any superset $\Pi_i' \supset \Pi_i$, we just have to note that $|K_i^{\Pi_i'}| \geq |K_i^{\Pi_i}|$ (because the overall number of non-zero counts can only increase when we include more parents), and $\alpha_{ij'}^{\Pi_i'}$ (for all j') are all less than 0.8349 (because the αs can only decrease when more parents are included), thus we can apply the very same reasoning to all supersets. ■

Theorem 9 provides a bound to discard parent sets without even inspecting them because of the non-increasing monotonicity of the employed bounding function when we increase the number of parents. As done for the BIC and AIC criteria, the idea is to check the validity of Theorem 9 every time the score of a parent set Π_i of X_i is about to be computed by taking the best score of any subset and testing it against the theorem (of course using only subsets that satisfy the structural constraints). Whenever possible, we discard Π_i and do not even look into all its supersets. Note that the assertion $\alpha_{ij} \leq 0.8349$ required by the theorem is not too restrictive, because as parent sets grow, as ESS is divided by larger numbers (it is an exponential decrease of the αs). Hence, the values α_{ij} become quickly below such a threshold. Furthermore, Π_i is also checked against Lemma 1 (although it does not help with the supersets). As we see later in the experiments, the practical size of the cache after the application of the properties is small even for considerably large networks, and both Lemma 1 and Theorem 9 help reducing the cache size, while Theorem 9 also help to reduce computations. Finally, we point out that Singh and Moore (2005) have already worked on bounds to reduce the number of parent sets that need to be inspected, but Theorem 9 provides a much tighter bound than their previous result, where the cut happens only after all $|K_{ij}^{\Pi_i}|$ go below two (or using previous terminology, when *configurations are pure*).

5. Constrained B&B algorithm

In this section we describe the branch-and-bound (B&B) algorithm used to find the best structure of the Bayesian network and comment on its complexity and correctness. The algorithm uses a B&B search where each case to be solved is a relaxation of a DAG, that is, the cases may contain cycles. At each step, a graph is picked up from a priority queue, and it is verified if it is a DAG. In such case, it is a feasible structure for the network and we compare its score against the best score so far (which is updated if needed). Otherwise, there must be a directed cycle in the graph, which is then broken into subcases by forcing some arcs to be absent/present. Each subcase is put in the queue to be processed (these subcases cover all possible subgraphs related to the original case, that is, they cover all possible ways to break the cycle). The procedure stops when the queue is empty. Note that every time we break a cycle, the subcases that are created are independent, that is, their sets of graphs are disjoint. We obtain this fact by properly breaking the cycles to avoid overlapping among subcases (more details below). This is the same idea as in the inclusion–exclusion principle of combinatorics employed over the set of arcs that formed the cycle and ensures that we never process the same graph twice, and also ensures that all subgraphs are covered.

The initialization of the algorithm is as follows:

- $C : (X_i, \Pi_i) \to \mathcal{R}$ is the cache with the scores for all the variables and their possible parent configurations. This is constructed using a queue and analyzing parent sets according to the properties of Section 4, which saves (in practice) a large amount of space and time. All the structural constraints are considered in this construction so that only valid parent sets are stored.

- $\mathcal{G}$ is the graph created by taking the best parent configuration for each node without checking for acyclicity (so it is not necessarily a DAG), and s is the score of $\mathcal{G}$. This graph is used as an upper bound to the best possible graph, as it is clearly obtained from a relaxation of the problem (the relaxation comes from allowing cycles).

- $\mathcal{H}$ is an initially empty matrix containing, for each possible arc between nodes, a mark stating that the arc must be present, or is prohibited, or is free (may be present or not). This matrix controls the search of the B&B procedure. Each branch of the search has a $\mathcal{H}$ that specifies the graphs that still must be searched within that branch.

- Q is a priority queue of triples $(\mathcal{G}, \mathcal{H}, s)$, ordered by s (initially it contains a single triple with $\mathcal{G}$, $\mathcal{H}$ and s as mentioned. The order is such that the peak contains always the triple of greatest s.

- $(\mathcal{G}_{best}, s_{best})$ keeps at any moment the best DAG and score found so far (s_{best} is initialized with $-\infty$). In fact, this best solution can be initialized using any inner approximation method. For instance, we use a procedure that guesses an ordering for the variable, then computes the global best solution for that ordering, and finally runs a hill climbing over the resulting structure. All these procedures are very fast (given the small size of the pre-computed cache that we obtain in the previous steps). A good initial solution may significantly reduce the search of the B&B procedure, because it may give a lower bound closer to the upper bound defined by the relaxation $(\mathcal{G}, \mathcal{H}, s)$.

The main loop of the B&B search is as follows:

- While Q is not empty, do

  1. Remove the peak $(\mathcal{G}_{cur}, \mathcal{H}_{cur}, s_{cur})$ of Q. If $s_{cur} \leq s_{best}$ (worse than an already known solution), then discard it and start the loop again.
  2. If $\mathcal{G}_{cur}$ is a DAG (it certainly satisfies all structural constraints, because all the elements in the cache do so), update $(\mathcal{G}_{best}, s_{best})$ with $(\mathcal{G}_{cur}, s_{cur})$ and start the loop again.
  3. Take a cycle of $\mathcal{G}_{cur}$ (one must exist, otherwise we would have not reached this step), namely $v = (X_{a_1} \to X_{a_2} \to \ldots \to X_{a_{q+1}})$, with $a_1 = a_{q+1}$. This can be computed by a single search in the graph.

4. For $y = 1, \ldots, q$, do
   - Mark on $\mathcal{H}_{cur}$ that the arc $X_{a_y} \to X_{a_{y+1}}$ is prohibited. This implies that the branch we are going to create will not have this cycle again.
   - Recompute $(\mathcal{G}, s)$ from $(\mathcal{G}_{cur}, s_{cur})$ such that the new parent set of $X_{a_{y+1}}$ in $\mathcal{G}$ complies with this new $\mathcal{H}_{cur}$. This is done by searching in the cache $C(X_{a_{y+1}}, \Pi_{a_{y+1}})$ for the best parent set. If there is a parent set in the cache that satisfies $\mathcal{H}_{cur}$, then include the triple $(\mathcal{G}, \mathcal{H}_{cur}, s)$ into Q.
   - Mark on $\mathcal{H}_{cur}$ that the arc $X_{a_y} \to X_{a_{y+1}}$ must be present and that the sibling arc $X_{a_{y+1}} \to X_{a_y}$ is prohibited, and continue the loop of step 4. (*This last step forces the branches that we are creating to be disjoint among each other.*)

There are two considerations to show the correctness of the method. First, we need to guarantee that all the search space is considered, even though we do not search through all of it. Second, we must ensure that the same part of the search space is not processed more than once, so we do not lose time and know that the algorithm will finish with a best global graph. The search is conducted over all possible graphs (not necessarily DAGs). The queue Q contains the subspaces (of all possible graphs) to be analyzed. A triple $(\mathcal{G}, \mathcal{H}, s)$ indicates, through $\mathcal{H}$, which is this subspace. $\mathcal{H}$ is a matrix containing an indicator for each possible arc. It says if an arc is allowed (meaning it might or might not be present), prohibited (it cannot be present), or demanded (it must be present) in the current subspace of graphs. Thus, $\mathcal{H}$ completely defines the subspaces. $\mathcal{G}$ and s are respectively the best graph inside $\mathcal{H}$ (note that $\mathcal{G}$ might have cycles) and its score value (which is an upper bound for the best DAG in this subspace).

In the initialization step, Q begins with a triple where $\mathcal{H}$ indicates that every arc is allowed,[5] so all possible graphs are within the subspace of the initial $\mathcal{H}$. The score s of $\mathcal{G}$ is compared against the best known score. Note that as $\mathcal{G}$ is the graph with the greatest score that respects $\mathcal{H}$, any other graph within the subspace defined by $\mathcal{H}$ will have worse score. Therefore, if s is less than the best known score, all this branch represented by $\mathcal{H}$ may be discarded (this is the *bound* step). Certainly no graph in that subspace will be worth, because their scores are less than s.

If $\mathcal{G}$ has score greater than s_{best}, then the graph $\mathcal{G}$ is checked for cycles, as it may or may not be acyclic (all we know is that $\mathcal{G}$ is a relaxed solution within the subspace $\mathcal{H}$). If it is acyclic, then $\mathcal{G}$ is the best known graph and the best score is updated. If $\mathcal{G}$ is cyclic, then we need to divide the space $\mathcal{H}$ into smaller subcases with the aim of removing the cycles of $\mathcal{G}$ (this is the *branch* step). Two characteristics must be kept by the branch step: (i) $\mathcal{H}$ must be fully represented in the subcases (so we do not miss any graph), and (ii) the subcases must be disjoint (so we do not process the same graph more than once). A possible way to achieve these two requirements is as follows: let the cycle $v = (X_{a_1} \to X_{a_2} \to \ldots \to X_{a_{q+1}})$ be the one detected in $\mathcal{G}$. We create q subcases such that

- The first subcase does not contain $X_{a_1} \to X_{a_2}$ (but may contain the other arcs of that cycle, that is, we do not prohibit the others).

5. In fact, the implementation may be smarter and set $\mathcal{H}$ with possible known restrictions of arcs, that is, those that are known to be demanded or prohibited by structural constraints may be included in the initial $\mathcal{H}$.

- The second case certainly contains $X_{a_1} \to X_{a_2}$, but $X_{a_2} \to X_{a_3}$ is prohibited (so they are disjoint because of the difference in the presence of the first arc).
- (And so on such that) The y-th case certainly contains $X_{a_{y'}} \to X_{a_{y'+1}}$ for all $y' < y$ and prohibits $X_{a_y} \to X_{a_{y+1}}$. This is done until the last element of the cycle.

This is the same idea as the inclusion–exclusion principle, but applied here to the arcs of the cycle. It ensures that we never process the same graph twice, and also that we cover all the graphs, as by the union of the mentioned sets we obtain the original $\mathcal{H}$. Because of that, the algorithm runs at most $\prod_i |C(X_i)|$ steps, where $|C(X_i)|$ is the size of the cache for X_i (there are not more ways to combine parent sets than that number). In practice, we expect the *bound* step to be effective in dropping parts of the search space in order to reduce the total time cost.

B&B can be stopped at any time and the current best solution as well as an upper bound for the global best score are available. This stopping criterion might be based on the number of steps, time and/or memory consumption, percentage of error (difference between the upper and lower bounds). This is an important property of this method. For example, if we are just looking for an improving solution, we may include in the loop an *if* to check if the current best solution is already better than some threshold, which would save computational time. Still, if we run it until the end, we are ensured to have a global optimum solution.

The algorithm can also be easily parallelized. We can split the content of the priority queue into many different tasks. No shared memory needs to exist among tasks if each one has its own version of the cache. The only data structure that needs consideration is the queue, which from time to time must be balanced between tasks. With a message-passing idea that avoids using process locks, the gain of parallelization is linear in the number of tasks. If run until it ends, the proposed method gives a global optimum solution for the structure learning problem.

Some particular cases of the algorithm are worth mentioning. If we fix an ordering for the variables such that all the arcs must link a node towards another non-precedent in the ordering (this is a common idea in many approximate methods), the proposed algorithm does not perform any branch, as the ordering implies acyclicity, and so the initial solution is already the best (for that ordering – the number of possible orderings is exponential in n). The performance would be proportional to the time to create the cache. Another important case is when one limits the maximum number of parents of a node. This is relevant for hard problems with many variables, as it would imply in a bound on the cache size.

6. Experiments

We perform experiments to show the benefits of the reduced cache and search space. Later we show some examples of the use of constraints.[6] First, we use data sets available at the UCI repository (Asuncion and Newman, 2007). Lines with missing data are removed and continuous variables are discretized over the mean into binary variables. The data sets are: *adult* (15 variables and 30162 instances), *breast* (10 variables and 683 instances), *car* (7 variables and 1728 instances) *letter* (17 variables and 20000 instances), *lung* (57 variables

6. The software is available online in the web address *http://www.ecse.rpi.edu/∼cvrl/structlearning.html*

	ESS	adult	breast	car	letter	lung	mush	nurse	wdbc	zoo
	0.1	6.2	0.0	0.1	3.7	1699.6	7.5	0.9	221.2	0.4
Memory	1	6.2	0.0	0.1	3.7	1150.1	5.9	0.8	204.6	0.4
(in MB)	10	6.3	0.0	0.1	3.8	812.3	5.4	0.7	206.2	0.3
	BIC	1.8	0.0	0.0	2.3	0.3	0.5	0.4	5.3	0.1
	0.1	89.3	0.0	0.0	429.4	2056	357.9	0.7	2891	1.7
Time	1	91.6	0.0	0.0	440.4	1398	278.7	0.7	2692	1.7
(in sec.)	10	91.6	0.0	0.0	438.1	1098	268.9	0.7	2763	1.7
	BIC	67.4	0.0	0.1	859.6	1.3	72.1	1.4	351	0.3
	0.1	$2^{17.4}$	$2^{10.5}$	$2^{8.8}$	$2^{20.1}$	$2^{30.8}$	$2^{24.0}$	$2^{11.2}$	$2^{27.9}$	$2^{19.8}$
Number	1	$2^{17.4}$	$2^{10.5}$	$2^{8.8}$	$2^{20.1}$	$2^{30.2}$	$2^{23.6}$	$2^{11.2}$	$2^{27.8}$	$2^{19.7}$
of Steps	10	$2^{17.4}$	$2^{10.4}$	$2^{8.8}$	$2^{20.1}$	$2^{29.8}$	$2^{23.5}$	$2^{11.2}$	$2^{27.9}$	$2^{19.6}$
	BIC	$2^{14.8}$	$2^{7.3}$	$2^{8.4}$	$2^{19.0}$	$2^{15.4}$	$2^{17.1}$	$2^{10.9}$	$2^{20.7}$	$2^{13.1}$
Worst-case		$2^{17.9}$	$2^{12.3}$	$2^{8.8}$	$2^{20.1}$	$2^{31.1}$	$2^{26.5}$	$2^{11.2}$	$2^{28.4}$	$2^{20.1}$

Table 1: Memory, time and number of steps (local score evaluations) used to build the cache. Results for BIC and BDeu score with ESS varying from 0.1 to 10 are presented.

and 27 instances), mushroom (23 variables and 1868 instances, denoted by *mush*), nursery (9 variables and 12960 instances, denoted by *nurse*), Wisconsin Diagnostic Breast Cancer (31 variables and 569 instances, denoted by *wdbc*), *zoo* (17 variables and 101 instances). The number of categories per variables varies from 2 to dozens in some cases (we refer to UCI for further details).

Table 1 presents the used memory in MB (first block), the time in seconds (second block) and number of steps in local score evaluations (third block) for the cache construction, using the properties of Section 4. Each column presents the results for a distinct data set. In different lines we show results for BDeu with ESS equals to 0.1, 1, 10, and for BIC. The line *worst-case* presents the number of steps to build the cache without using Theorems 4 (for BIC/AIC) and 9 (for BDeu), which are the theorems that allow the algorithm to avoid computing every subset of parents. As we see through the log-scale in which they are presented, the reduction in number of steps has not been exponential, but still saves a good amount of computations (roughly half of the work). In the case of the BIC score, the reduction is more significant. In terms of memory, the usage clearly increases with the number of variables in the network (lung has 57 and wdbc has 31 variables).

The benefits of the application of these results imply in performance gain for many algorithms in the literature to learn Bayesian network structures, as long as they only need to work over the (already precomputed) small cache. In Table 2 we present the final cache characteristics, where we find the most attractive results, for instance, the small cache sizes when compared to the worst case. The first block contains the maximum number of parents per node (averaged over the nodes, and the actual maximum between parenthesis). The worst-case is the total number of nodes in the data set minus one, apart from *lung* (where we have set a limit of at most six parents) and *wdbc* (with at most eight parents). The second block shows the cache size for each data set and distinct values of ESS. We also show the results of the BIC score and the worst-case values for comparison. We see that

	ESS	adult	breast	car	letter	lung	mush	nurse	wdbc	zoo
Max.	0.1	2.1(4)	1.0(1)	0.7(1)	4.5(5)	0.1(2)	4.1(5)	1.2(3)	1.3(2)	1.4(3)
Number	1	2.4(4)	1.0(1)	1.0(2)	5.2(6)	0.4(2)	4.4(7)	1.7(3)	1.7(3)	1.9(4)
of Parents	10	3.3(5)	1.0(1)	1.9(2)	5.9(6)	3.0(4)	4.8(8)	2.1(3)	3.1(4)	3.4(4)
	BIC	2.8(5)	1.0(1)	1.3(2)	6.3(7)	2.1(3)	4.1(4)	1.8(3)	2.7(3)	2.8(3)
Worst-case		14.0	9.0	6.0	16.0	6.0*	22.0	8.0	8.0*	16.0
Final Size	0.1	$2^{4.2}$	$2^{1.5}$	$2^{1.1}$	$2^{8.2}$	$2^{0.2}$	$2^{8.5}$	$2^{1.9}$	$2^{3.6}$	$2^{3.3}$
of the	1	$2^{4.8}$	$2^{1.9}$	$2^{1.6}$	$2^{9.0}$	$2^{0.8}$	$2^{8.9}$	$2^{2.4}$	$2^{4.9}$	$2^{4.4}$
Cache	10	$2^{6.3}$	$2^{3.3}$	$2^{3.0}$	$2^{10.5}$	$2^{10.7}$	$2^{9.8}$	$2^{3.5}$	$2^{12.1}$	$2^{8.9}$
	BIC	$2^{9.3}$	$2^{4.7}$	$2^{4.5}$	$2^{15.3}$	$2^{11.5}$	$2^{13.0}$	$2^{5.6}$	$2^{12.9}$	$2^{10.9}$
Worst-case		$2^{17.9}$	$2^{12.3}$	$2^{8.8}$	$2^{20.1}$	$2^{31.1*}$	$2^{26.5}$	$2^{11.2}$	$2^{28.4*}$	$2^{20.1}$
Implied	0.1	$2^{54.1}$	$2^{13.3}$	$2^{6.3}$	$2^{129.0}$	$2^{8.2}$	$2^{175.7}$	$2^{11.6}$	$2^{90.3}$	$2^{39.3}$
Search	1	$2^{62.1}$	$2^{17.1}$	$2^{8.3}$	$2^{144.8}$	$2^{33.1}$	$2^{186.0}$	$2^{15.4}$	$2^{132.7}$	$2^{60.3}$
Space	10	$2^{91.6}$	$2^{33.2}$	$2^{20.6}$	$2^{176.1}$	$2^{612.0}$	$2^{221.8}$	$2^{27.3}$	$2^{375.1}$	$2^{150.7}$
(approx.)	BIC	2^{71}	2^{23}	2^{10}	2^{188}	2^{330}	2^{180}	2^{17}	2^{216}	2^{111}
Worst-case		2^{210}	2^{90}	2^{42}	2^{272}	2^{1441*}	2^{506}	2^{72}	2^{727*}	2^{272}

Table 2: Final cache characteristics: maximum number of parents (average by node; between parenthesis is presented the actual maximum number), actual cache size, and (approximate) search space implied by the cache. Worst-cases are presented for comparison (those marked with a star are computed using the constraint on the number of parents that was applied to *lung* and *wdbc*). Results of BIC and BDeu with ESS from 0.1 to 10 are presented.

the actual cache size is smaller (in orders of magnitude) than the worst case situation. It is also possible to analyze the search space reduction implied by these results by looking the implications to the search space of structure learning. We must point out that by search space we mean all the possible combinations of parent sets for all the nodes. Eventually some of these combinations are not DAGs, but are still being counted. However, there are two considerations: (i) the precise counting problem is harder to solve (in order to give the exact search space size), and (ii) many structure learning algorithms run over more than only DAGs, because they need to look at the graphs (and thus combinations of parents) to decide if they are acyclic or not. In these cases, the actual search space is not simply the set of possible DAGs, even though the final solution will be a DAG. Still, some algorithms might do a better job by using other ideas of searching for the best structure instead of looking to possible DAGs, which might imply in a smaller worst case complexity (for instance, the dynamic programming method runs over subsets of variables, which are in number 2^n).

An expected but important point to emphasize is the correlation of the prior with the time and memory to build the cache. It would be expected that, as larger ESS (and thus the prior towards the uniform) as slower and more memory consuming is the method. That is because smoothing the different parent sets by the stronger prior makes harder to see large differences in scores, and consequently the properties that would reduce the cache size are less effective. However, this is not quite evident from the results, where the relation between ESS and time/memory is not clear. Yet it must be noted that the two largest data sets in terms of number of variables (*lung* and *wdbc*) were impossible to be processed without setting up other limits such as maximum number of parents or maximum number

		B&B			DP		OS		HC	
	network	Score	gap	time	score	time	score	time	score	time
BIC	adult	-286902.8	5.5%	150.3	0.0%	0.77	0.1%	0.17	0.5%	0.30
	breast	-8254.8	0.0%	0.01	0.0%	0.01	0.0%	0.01	0.0%	0.00
	car	-13100.5	0.0%	0.01	0.0%	0.01	0.0%	0.01	0.2%	0.00
	letter	-173716.2	8.1%	574.1	-0.6%	22.8	1.0%	0.75	3.7%	0.30
	lung	-1146.9	2.5%	907.1	*Fail*	*Fail*	1.0%	0.13	0.7%	0.05
	mushroom	-12834.9	15.3%	239.8	*Fail*	*Fail*	1.0%	0.12	4.8%	0.05
	nursery	-126283.2	0.0%	0.04	0.0%	0.04	0.0%	0.04	0.03%	0.06
	wdbc	-3053.1	13.6%	333.5	*Fail*	*Fail*	0.8%	0.13	0.9%	0.02
	zoo	-773.4	0.0%	5.2	0.0%	3.5	1.0%	0.03	0.6%	0.00
ESS=0.1	adult	-288591.2	0.0%	92.1	0.0%	0.75	0.1%	0.21	0.3%	0.32
	breast	-8635.1	0.0%	0.02	0.0%	0.01	0.0%	0.01	0.0%	0.00
	car	-13295.0	0.0%	0.01	0.0%	0.00	0.0%	0.00	0.1%	0.01
	letter	-181941.5	5.7%	375.75	-0.1%	7.6	0.1%	0.27	2.1%	0.27
	lung	-1731.9	0.0%	0.22	*Fail*	*Fail*	0.0%	0.11	0.0%	0.05
	mushroom	-12564.2	14.7%	382.4	*Fail*	*Fail*	0.2%	0.15	5.3%	0.05
	nursery	-126660.4	0.0%	0.06	0.0%	0.04	0.0%	0.04	0.1%	0.06
	wdbc	-3558.6	4.4%	494.1	*Fail*	*Fail*	1.4%	0.05	1.3%	0.01
	zoo	-1024.5	0.0%	0.09	0.0%	3.1	0.8%	0.01	1.0%	0.00
ESS=1	adult	-286695.2	4.5%	203.0	0.0%	0.76	0.1%	0.22	0.3%	0.34
	breast	-8254.3	0.0%	0.02	0.0%	0.01	0.0%	0.01	0.0%	0.00
	car	-13145.3	0.0%	0.01	0.0%	0.00	0.0%	0.00	0.05%	0.00
	letter	-178635.2	6.7%	520.2	-0.7%	9.9	0.0%	0.34	2.1%	0.27
	lung	-1249.7	0.0%	0.61	*Fail*	*Fail*	0.1%	0.12	0.1%	0.05
	mushroom	-12097.0	16.7%	381.5	*Fail*	*Fail*	0.2%	0.19	4.2%	0.05
	nursery	-126212.7	0.0%	0.06	0.0%	0.04	0.0%	0.04	0.1%	0.05
	wdbc	-3175.9	11.2%	471.1	*Fail*	*Fail*	0.7%	0.06	1.0%	0.02
	zoo	-794.1	0.0%	1.4	0.0%	3.4	1.1%	0.02	8.7%	0.00
ESS=10	adult	-285014.5	11.8%	213.8	-0.1%	0.88	0.04%	0.24	0.5%	0.33
	breast	-8130.2	0.0%	0.04	0.0%	0.01	0.0%	0.00	0.3%	0.00
	car	-13038.6	0.0%	0.03	0.0%	0.00	0.0%	0.00	0.03%	0.00
	letter	-174111.8	8.7%	1250	-0.4%	22.3	0.1%	0.84	1.8%	0.32
	lung	-957.2	11.7%	2118	*Fail*	*Fail*	3.3%	1.38	2.3%	0.1
	mushroom	-11924.0	22.7%	587.8	*Fail*	*Fail*	0.1%	0.43	2.4%	0.07
	nursery	-125846.5	0.0%	0.14	0.0%	0.04	0.0%	0.04	0.1%	0.06
	wdbc	-2986.2	22.2%	1938	*Fail*	*Fail*	0.6%	2.8	1.4%	0.23
	zoo	-697.2	13.2%	367.7	-0.3%	5.0	1.4%	0.1	0.9%	0.00

Table 3: Comparison of scores among B&B, DP, OS and HC. *Fail* means that it could not solve the problem within 10 million steps or because of memory limit (4GB). DP, OS and HC scores are in percentage w.r.t. the score of B&B (positive means worse than B&B and negative means better). Each entry with a 0.0% means that the result, in that instance, was exactly equal to the B&B result (in terms of the score). Times are given in seconds.

of free parameters in the node (we have not used any limit for the other data sets). We used an upper limit of six parents per node for *lung* and eight for *wdbc*. This situation deserves further study so as to clarify whether it is possible to run these computations on large data sets and large ESS. It might be necessary to find tighter bounds if at all possible, that is, stronger results than Theorem 9 to discard unnecessary score evaluations earlier in the computations. Nevertheless, the main goal of this present work is not to study the impact of ESS on learning, but to present properties that improve the performance of learning methods.

In Table 3, we show results of four distinct algorithms: the B&B described in Section 5, the dynamic programming (DP) idea of Silander and Myllymaki (2006), the hill-climbing (HC) method starting with an empty structure, and an algorithm that picks variable orderings randomly and then find the best structure such that all arcs link a node towards another that is not precedent in the ordering. This algorithm (named OS) is similar to K2 algorithm with random orderings, but it is always better because a global optimum is found for each ordering. Note that OS performs better than HC in almost all test cases. We have chosen to analyze the BIC scores (given that the properties have provided greater reduction in the search space in this case) and BDeu with ESS equals to 0.1, 1 and 10. It is clear from the results of ESS equals to 10 that the B&B procedure struggles with very large search spaces, and the same might happen for even larger ESS.

The scores obtained by each algorithm (in percentage against the value obtained by B&B) and the corresponding time are shown in Table 3 (excluding the cache construction). A limit of ten million steps is given to each method (steps here are considered as the number of queries to the cache). It is also presented the reduced space where B&B performs its search, as well as the maximum gap of the solution. This gap is obtained by the relaxed version of the problem. We can guarantee that the global optimal solution is within this gap (even though the solution found by the B&B may already be the best, as it happens, for example, in the first line of the table). With the reduced cache presented here, finding the best structure for a given ordering is very fast, so it is possible to run OS over millions of orderings in a short period of time. Some additional comments are worth. DP could not solve *wdbc* or *lung* even without the limit in number of steps, because it has exhausted 16GB of memory. Hence, we cannot expect to obtain answers in larger cases. However, it is clear that (in a worst case sense) the number of steps of DP is smaller than that of B&B, and this behavior can be seen in data sets with small number of variables. Nevertheless, B&B eventually bounds some regions without processing them, provides an upper bound at each iteration, and does not suffer from memory exhaustion as DP. It is true that B&B also uses memory increasingly if there are not good bounds, but this case can be tackled by (automatically) switching between B&B and a depth-first search.[7] This makes the method applicable even to very large settings. When n is large (more than 35), DP will not finish in reasonable time, and hence will not provide any solution, while B&B still gives an approximation and a bound to the global optimum. About OS, if we sample even more orderings, then its results improve and the global optimum is found also for *adult* and *mushroom* sets. Still, OS provides no guarantee or estimation about how far is the global optimum (here we know that the optimum has been achieved because of the solution of the exact methods). It is worth noting that both DP and OS are also benefited by the

7. Our implementation is able to switch between breath-first and depth-first searches, but this behavior was not used in the experiments of this paper.

smaller cache. Although we are discussing only four algorithms, performance gain from the application of the properties in other algorithms is expected as well.

network	time(s)	cache size	space
adult	0.26	114	2^{39}
car	0.01	14	$2^{6.2}$
letter	0.32	233	2^{61}
lung	0.26	136	2^{51}
mushroom	0.71	398	2^{88}
nursery	0.06	26	2^{12}
wdbc	361.64	361	2^{99}
zoo	8.4	1697	2^{111}

Table 4: B&B procedure learning TANs using BIC. Time (in seconds) to find the global optimum, cache size (number of stored scores) and (reduced) space for the B&B search.

The last part of this section is dedicated to some test cases with constraints. Table 4 shows the results when we employ constraints to force the final network to be a Tree-augmented Naive Bayes. Here the class variable is isolated in the data set and constraints are included as described in Section 3. Note that the cache size, the search space and consequently the time to solve the problems have substantially decreased. Finally, Table 5 has results for random data sets with predefined number of nodes and instances using the BIC score. A randomly created Bayesian network with at most $3n$ arcs (where n is the number of nodes) is used to sample the data. Because of that, we are able to generate random structural constraints that are certainly valid for this *true* Bayesian network (approximately n constraints for each case). The table contains the total time to run the problem and the size of the cache, together with the results when using constraints. Note that the code was run in parallel with a number of tasks equals to n, otherwise an increase by a factor of n must be applied to the results in the table. Each line contains the mean and standard deviation of ten executions (using random generated networks) for time and cache size with and without constraints (using the same data sets in order to compare them). We can see that the gain is recurrent in all cases. The B&B method was able to find a global optimal solution in all but the cases with one hundred nodes, where it has achieved an approximate solution with error always less than 0.1% (this amounts to 40% of the test cases with 100 nodes). We point out that the other exact method we have analyzed based on dynamic programming cannot deal with such large networks because of both memory and time costs. However, we are not considering the improvement in accuracy when using constraints, but just the computational gain. It is not trivial to measure the quality of a learned structure, because the target of the methods is the underlying probability distribution, and distinct structures may lead to good results in fitting such distribution. For instance, comparing number of matching arcs has only meaning if one is interested in the structure by itself, and not in the fitness of the underlying distribution. This topic deserves attention, but it would bring us far from the focus of this study.

nodes(n)/	unconstrained				constrained			
	time(sec)		cache size		time(sec)		cache size	
instances	mean	std.dev.	mean	std.dev.	mean	std.dev.	mean	std.dev.
30/100	0.07	0.02	49.6	9.1	0.04	0.01	44.3	8.98
30/500	3.70	1.18	75.6	16.6	2.33	0.73	61.4	17.7
50/100	0.31	0.08	77.9	9.6	0.20	0.04	66.1	6.71
50/500	37.1	10.8	102.5	23.0	23.2	6.86	83.0	17.7
70/100	1.91	0.82	127.5	18.1	0.97	0.32	108.3	13.6
70/500	293.3	99.5	137.3	22.2	176.3	62.6	111.8	14.5
100/100	85.0	29.3	253.4	27.7	4.44	1.06	199.5	21.1
100/500	2205.6	534.4	204.6	32.1	1414.8	419.2	168.0	21.3

Table 5: Results on ten data sets per line generated from random networks. Both mean and standard deviation of time to solve (with an upper limit of 20 million steps) and size of the cache (in number of scores) are presented for the *normal* unconstrained case and for the constrained cases (over the same data sets).

7. Conclusions

This paper describes novel properties of decomposable score functions to learn Bayesian network structure from data. Such properties allow the construction of a cache with all possible local scores of nodes and their parents without large memory consumption, which can later be used by searching algorithms. For instance, memory consumption was a bottleneck for some algorithms in the literature, see for example Parviainen and Koivisto (2009). This implies in a considerable reduction of the search space of graphs without losing the global optimal structure, that is, it is ensured that the overall best graph remains in the reduced space. In fact the reduced memory and search space potentially benefits many structure learning methods in the literature, and we have conducted experiments with some of them.

An algorithm based on a branch-and-bound technique is described, which integrates structural constraints with data. The procedure guarantees global optimality with respect the score function. It is an any-time procedure in the sense that, if stopped early, it provides the best current solution found so far and a maximum error of such solution. This is specially important if one wants to integrate it with an expectation–maximization (EM) method to treat incomplete data sets, and such characteristic is usually not present in other exact structure learning methods. Inside the EM method, the global structure learning procedure ensures that the maximization step is never trapped by a local solution, and the anytime property allows the use of a generalized EM idea to reduce considerably the computational cost.

Because of the properties and the characteristics of the B&B method, it is more efficient than dynamic programming state-of-the-art exact methods for large domains. We show through experiments with randomly generated data and public data sets that problems with up to 70 nodes can be exactly processed in reasonable time, and problems with 100 nodes are handled within a small worst case error. Dynamic programming methods are able to treat less than 35 variables. Described ideas may also help to improve other approximate methods and may have interesting practical applications. We show through experiments with public data sets that requirements of memory are small, as well as the resulting reduced search space. Of course we do not expect to exactly solve problems for considerably large networks, still the paper makes a relevant step towards solving larger instances. We can summarize the comparison with the dynamic programming idea as follows: if the problem has few variables, dynamic programming is probably the fastest method (the branch-and-bound method will also be reasonably fast); if the problem has medium size, the branch-and-bound method might solve it exactly (dynamic programming will mostly fail to answer); finally, if the problem is large, the branch-and-bound method will eventually give an approximation (and its worst-case error), while the standard dynamic programming idea will fail.

There is certainly much further to be done. One important question is whether the bounds of the theorems in Section 4 (more specifically Theorem 9) can be improved or not. We are actively working on this question. Furthermore, the experimental analysis can be extended to further clarify the understanding of the problem, for instance how the ESS affects the results. It is clear that, for considerably large domains, none of the exact methods are going to suffice by themselves. Besides developing ideas and algorithms for dealing with large domains, the comparison of structures and what define them to be good is an important topic. For example, accuracy of the generated networks can be evaluated with real data. On the other hand, it does not ensure that we are finding the true links of the underlying structure, but a somehow similar graph that produces a close joint distribution. For that, one could use generated data and compare the structures against the one data were generated from it. A study on how the properties may help fast approximate methods is also a desired goal.

Acknowledgments

This work is supported in part by the grant W911NF-06-1-0331 from the U.S. Army Research Office, and by the *Computational Life Sciences (CLS)* program phase II, canton Ticino, Switzerland.

References

H. Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton Univ. Press, 2006.

A. Asuncion and D. J. Newman. UCI machine learning repository. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007.

R. Bouckaert. Properties of bayesian belief network learning algorithms. In *10th Conference on Uncertainty in Artificial Intelligence*, pages 102–109, San Francisco, CA, 1994. Morgan Kaufmann.

W. Buntine. Theory refinement on Bayesian networks. In *7th Conference on Uncertainty in Artificial Intelligence*, pages 52–60, San Francisco, CA, 1991. Morgan Kaufmann.

D. M. Chickering, D. Heckerman, and C. Meek. Large-Sample Learning of Bayesian Networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

C. P. de Campos, Z. Zeng, and Q. Ji. Structure learning of Bayesian networks using constraints. In *26th International Conference on Machine Learning*, pages 113–120, Montreal, June 2009. Omnipress.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

T. Jaakkola, D. Sontag, A. Globerson, and M. Meila. Learning Bayesian Network Structure using LP Relaxations. In *13th International Conference on Artificial Intelligence and Statistics*, pages 358–365, 2010.

M. Koivisto. Advances in exact bayesian structure discovery in bayesian networks. In *22nd Conference on Uncertainty in Artificial Intelligence*, pages 241–248, AUAI Press, 2006.

M. Koivisto and K. Sood. Exact Bayesian Structure Discovery in Bayesian Networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

K. Kojima, E. Perrier, S. Imoto, and S. Miyano. Optimal search on clustered structural constraint for learning Bayesian network structure. *Journal of Machine Learning Research*, 11:285–310, 2010.

S. Ott and S. Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003.

P. Parviainen and M. Koivisto. Exact structure discovery in bayesian networks with less space. In *25th Conference on Uncertainty in Artificial Intelligence*, 2009.

F. Pernkopf and J. Bilmes. Discriminative versus generative parameter and structure learning of bayesian network classifiers. In *22nd international conference on Machine learning*, pages 657–664, New York, NY, 2005. ACM.

E. Perrier, S. Imoto, and S. Miyano. Finding optimal bayesian network given a super-structure. *Journal of Machine Learning Research*, 9:2251–2286, 2008.

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

T. Silander and P. Myllymaki. A simple approach for finding the globally optimal bayesian network structure. In *22nd Conference on Uncertainty in Artificial Intelligence*, pages 445–452, Arlington, Virginia, 2006. AUAI Press.

A. P. Singh and A. W. Moore. Finding optimal bayesian networks by dynamic programming. Technical report, Carnegie Mellon University, 2005. CMU-CALD-05-106.

P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. Springer-Verlag, New York, 1993.

J. Suzuki. Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique. In *13th International Conference on Machine Learning*, pages 462–470, 1996.

M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *21st Conference on Uncertainty in Artificial Intelligence*, pages 584–590, 2005, AUAI Press.

I. Tsamardinos, L. E. Brown, and C. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, 2006.

A Factorization Approach to Evaluating Simultaneous Influence Diagrams

Weihong Zhang and Qiang Ji, *Senior Member, IEEE*

***Abstract*—Evaluating an influence diagram (ID) is a challenging problem because its complexity increases exponentially in the number of decision nodes in the diagram. In this paper, we examine the problem for a special class of IDs where multiple decisions must be made simultaneously. We describe a brief theory that factorizes out the computations common to all policies in evaluating them. Our evaluation approach conducts these computations once and uses them across all policies. We identify the ID structures for which the approach can achieve savings. We show that the approach can be used to efficiently recompute the optimal policy of an ID when its structure or parameters change. Finally, we demonstrate the superior performance of the approach by simulation studies and a military planning example.**

***Index Terms*—Algorithm, decision making under uncertainty, graphical model, influence diagram (ID), military analysis.**

I. Introduction

An INFLUENCE diagram (ID) is a plausible graphical model for decision making under uncertainty [1]. An ID comprises of decision nodes, random nodes, value nodes, and the probabilistic relations among these nodes. An ID is a more compact representation of a decision tree, which is a simple tool for decision analysis [2].

Given an ID, a policy prescribes an action choice for each decision node. Evaluating a policy is to compute the expected value of the ID under the policy. Evaluating an ID is to find the optimal policy that maximizes the expected value of the ID. A generic approach to evaluating an ID has to enumerate all policies, compare the expected utilities under them, and choose the optimal one. However, the number of policies grows exponentially with the number of decision nodes. This renders the approaches for general ID evaluation very inefficient and infeasible for large IDs. Consequently, it is advisable to study efficient algorithms for special IDs.

Most of the previous approaches assume that there exists a linear ordering among the decision nodes. This ordering implies that the choice of a decision node is known to the decision maker when he/she chooses the actions for the successive decision nodes (e.g., see [13]). For a decision node, this linear ordering usually can be exploited to decompose the ID into one fraction prior to the node and the other fraction posterior to the node. The choice for the decision node can be made using the fraction posterior to the node. The procedure repeats for each decision node.

In this paper, we examine the ID evaluation problem for a special class of IDs in which decision nodes have no parents. Essentially, an ID with this property assumes no precedence relationship among decision nodes. In other words, one has to determine the choices for all decision nodes simultaneously. For this reason, such an ID is said to be simultaneous. The simultaneity assumption prevails in real-world problem domains. For instance, a military planner must select among a number of available actions to achieve his/her overall goal success; a business owner must consider multiple elements in order to maximize his/her monetary profit.

In evaluating a simultaneous ID, we exploit the assumption and divide the ID into two fractions, calling them the upstream and downstream. Roughly, the upstream consists of decision nodes and their children nodes through which the decisions propagate their impacts on the ID. Informally, these child nodes are called interface nodes. The downstream consists of the interface nodes and their succeeding nodes. We present a representation theorem, showing that the expected value of a value node under a policy can be represented as the sum of some intermediate quantities weighted by the probabilities determined by the policy. These intermediate quantities involve only the downstream. The factorization approach we proposed computes them once but uses them across all policies. The computational gain brought by the approach depends on the size of the downstream. Usually, larger downstream size implies more savings.

We organize the paper as follows. In the next section, we discuss related work to this research. We then introduce IDs and the evaluation problem. In Section IV, we describe the representation theorem and develop the factorization approach. In Section V, we discuss two extensions of the approach: how it can be adapted to network structure/parameter changes and how it can be used in planning over time. We report empirical results on simulation studies and a military planning example in Section VI. Finally, we conclude the paper in Section VII.

II. Related Work

Since IDs were introduced by Howard and Matheson [1], a variety of approaches have been proposed to find the optimal policy of a given ID. To mitigate the exponential growth

Manuscript received February 19, 2004; revised July 20, 2004 and December 22, 2004. The work was supported in part by the Air Force Office of Scientific Research (AFOSR) under Grant F49620-03-1-0160. Part of the work for this project was also supported by the Air Force Research Laboratory (AFRL)/Rome summer visiting faculty program. This paper was recommended by Associate Editor Yang.

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590 USA.

Digital Object Identifier 10.1109/TSMCA.2005.855753

problem of the policy number in the number of decision nodes, researchers have studied several special ID classes and proposed efficient approaches exploiting their specific problem characteristics. We give a brief survey of these IDs and their solutions.

A. Regular and No-Forgetting IDs

To some extent, most IDs that have been studied assume a precedence ordering of the decision nodes. A regular ID assumes that there is a directed path containing all decision nodes; a no-forgetting ID assumes that each decision node and its parents are also parents of the successive decision nodes; and a stepwise decomposable ID assumes that the parents of each decision node divide the ID into two separate fractions. These assumptions are different from ours, which requires the actions to be chosen simultaneously. There exist direct and indirect approaches evaluating a regular no-forgetting ID. A direct approach works on the ID and evaluates it directly. Shachter [3] proposed an algorithm that evaluates an ID by applying a series of value-preserving reductions. A value-preserving reduction is an operation that can transform an ID into another one with the same expected value. Specifically, Shachter identified the following four reductions arc reversal, barren-node removal, random-node removal, and decision-node removal. An indirect approach first transforms an ID into an intermediate structure whose optimal policy (or value) remains the same as in the original ID. It then evaluates the intermediate structure and obtains the optimal policy. For instance, Howard and Matheson discussed a way to transform an ID into a decision-tree network and to compute an optimal policy from the decision tree. In transforming an ID into a decision-tree network, a basic operation is arc reversal [1], [3]. Since a no-forgetting ID must be stepwise decomposable, stepwise decomposability is more general than no-forgetting.

In most ID evaluation approaches, the ordering of decision nodes is an important information source in decision making and therefore, is exploited to evaluate the optimal decision for decision nodes [4]–[6]. A stepwise decomposable ID can be evaluated by a divide-and-conquer approach. The approach deals with one decision node at a time [7]. For each decision node, its parental set separates an ID into two parts—a body and a tail. The tail is a simple ID with only one decision node. The body's value node is a new value node whose value function is obtained by evaluating the tail. In evaluating a stepwise decomposable ID, the approach begins with a leaf decision node and repeats the decomposition/evaluation procedure for the preceding decision nodes. In evaluating the tail with only one single decision node, the problem is reduced to that of computing posterior probabilities in a Bayesian network. Hence, the approach uses probabilistic inference techniques to evaluate an ID. Cooper [8] initiated the research in this direction. He gave a recursive formula for computing the maximal expected utilities and optimal policies of IDs. Shachter and Peot [9] showed that the problem of ID evaluation can be reduced to a series of probabilistic inferences. Zhang [13] described an algorithm that induces much easier probabilistic inferences than those in [8] and [9].

B. Partial IDs

There also exists research work that relaxed the regularity or no-forgetting assumption. The specific ID types include partial IDs, unconstrained IDs and limited memory ID (LIMID), which is a compact representation of IDs. A partial ID is an ID that allows a non-total ordering of decision nodes [10]. Because the solution to a partial ID depends on the temporal ordering of the decisions, it is of interest to find the conditions identifying a class of partial IDs whose solution is independent of the legal evaluation ordering. Based on the concept of d-connectivity, Nielsen and Jensen presented an algorithm determining whether or not a partial ID represents well-defined scenarios, and they also addressed the problem of whether all admissible orderings yield the same optimal strategy.

An unconstrained ID is an ID where the order of decision nodes and the observable random nodes is not determined [11]. For an unconstrained ID, it is of interest to determine the order of decision nodes and information on which set of nodes is necessary for decision making in a decision node. For this purpose, a set of rules have been developed in order to determine the choice of the next decision node, given the current information. Such a decision choice may be dependent on the specific information from the past.

Another recently proposed ID is called LIMID, which violates the no-forgetting assumption [12]. In contrast to the regular and no-forgetting assumption, the assumption behind a LIMID is that only requisite information for the computation of optimal policies is depicted in the graphical representation. Two properties pertaining to LIMIDs are: 1) any ID can be converted to a LIMID; and 2) the converted LIMID is more compact than the original ID in the sense that only requisite information is depicted in the LIMID for computing an optimal policy. By these properties, one may convert an ID to its LIMID version and solve the LIMID instead of the original ID. This optimal policy is also optimal in the original ID. The algorithm solving a LIMID exploits the fact that the entire decision problem can be partitioned into a set of smaller decision problems, each of which has one decision node only. This is analogous to the divide-and-conquer approach [13].

C. Simultaneous IDs

From its root definition, an ID does not impose a precedence ordering of the decision nodes. As an example, there are military applications that need to choose multiple actions simultaneously. A simultaneous ID is suitable for this situation. We exploit this assumption and divide a simultaneous ID into the upstream and the downstream fractions. The decomposition takes the random and value nodes as interface nodes between the upstream and the downstream. The computations involving the downstream fraction can be precomputed and reused across all policies in evaluating them. This computation-sharing schema can greatly accelerate the procedure of finding the optimal policy for a given ID, as indicated in our theoretical and empirical analysis.

Technically, the factorization approach has some conceptual similarities to the probabilistic inference-based algorithm [13].

Both algorithms divide the ID into two fractions. However, there are apparent differences. In [13], the separation of an ID relies on a single decision node. With respect to a decision node, roughly, the body contains the predecessors of the decision nodes, while the tail contains the successors. The choice of the decision node is evaluated by the tail part. This is quite different from our factorization approach, where the separation relies on the set of interface nodes. The set of the interface nodes separates an ID into two fractions: roughly, the upstream contains the predecessors of all interface nodes, while the downstream contains the successors. This difference in solution techniques stems from the difference in assumption—the probabilistic inference algorithm works with a regular ID that specifies a linear order among decision nodes, whereas the factorization algorithm works with a simultaneous ID that assumes no ordering among decision nodes.

III. Influence Diagram

Mathematically, an ID $\mathcal{I}$ is a directed acyclic graph consisting of three types of nodes and the links among these nodes [1].

1) Its node set is partitioned into a set of random nodes $\mathcal{Y}$, a set of decision nodes $\mathcal{X}$, and a set of value nodes $\mathcal{U}$. A value node cannot have children. The links characterize the conditional dependence among the nodes in the ID. Specifically, links to a random node indicate the probabilistic dependence of the node on its parents; links to a decision node indicate the information available to the planner at the time the planner must choose a decision for it; and links to a value node indicate the functional dependencies.

   We will adopt the following notational conventions. We will use bold-typed letters such as $\mathcal{Z}$ to denote a set of variables and capital letters such as Z to denote a variable in the set. Each random or decision node Z is associated with a set Ω_Z, denoting the set of its possible states. The set Ω_Z is called the domain of node Z. An element in Ω_Z is denoted by a low-case letter z. For any node Z, we use $\pi(Z)$ to denote its parent set. For any subset $\mathcal{Z}' \subset \mathcal{Y} \cup \mathcal{X}$, we use $\Omega_{\mathcal{Z}'}$ to denote the Cartesian product $\Pi_{Z \in \mathcal{Z}'} \Omega_Z$. For convenience, we shall interchangeably use a node and a variable. Without loss of generality, we assume that all the nodes are binary throughout this paper.
2) For each decision or random node Z, given an assignment of $\pi(Z)$, the distribution $P(Z|\pi(Z))$ specifies the probability of Z being in each state of the node Z. Such a distribution is called a conditional probability table (CPT) in the case that the domain of the variable Z is a finite set.
3) For each value node U, g_U is a value function $g_U : \Omega_{\pi(U)} \rightarrow R$, where R denotes the set of the real numbers.

To avoid unnecessary notations, we define the (optimal) policy concept only for a simultaneous ID.[1] A policy, denoted by δ, specifies one action choice for each decision node in $\mathcal{X}$. Hence, a policy δ can be denoted by $(\delta_1, \ldots, \delta_n)$, where δ_i belongs to the domain of X_i for each i.

[1]For general IDs, the definition of an (optimal) policy can be found in, e.g., [6].

Given a policy δ, a probability P_δ can be defined over the random nodes and decision nodes as follows:

$$P_\delta(\mathcal{Y}, \mathcal{X}) = \Pi_{Y \in \mathcal{Y}} P\left(Y|\pi(Y)\right) \Pi_{i=1}^{n} P_\delta(X_i) \tag{1}$$

where $P(Y|\pi(Y))$ is specified in the definition of $\mathcal{I}$, while $P_\delta(X_i)$ is equal to 1.0 if $X_i = \delta_i$, and 0.0 otherwise.

The expectation of the value node U under policy δ, denoted by $E_\delta[U]$, is defined as

$$E_\delta[U] = \sum_{\pi(U)} P_\delta\left(\pi(U)\right) g_U\left(\pi(U)\right). \tag{2}$$

The expected value E_δ of $\mathcal{I}$ under the policy δ is the sum $E_\delta[U]$ over all value nodes U in $\mathcal{U}$, i.e.,

$$E_\delta = \sum_{U \in \mathcal{U}} E_\delta[U]. \tag{3}$$

For simplicity, E_δ is also called the expected value of policy δ. Evaluating a policy δ means to compute its expected value. The maximum of E_δ over all policies is the optimal (expected) value of $\mathcal{I}$. An optimal policy is the policy that achieves the optimal expected value. To evaluate an ID is to find an optimal policy and to compute its optimal expected value.

IV. The Factorization Approach

In this section, we describe the representation theorem and the factorization approach.

A. The Idea

From its definition, an ID is a network structure consisting of decision nodes, random nodes, and value nodes. Among them, in determining the expected value of the ID, a decision node plays a different role from a random or a value node. The choices of a decision node can affect the expected value of the ID through changing the CPTs of its child random nodes, or through changing the value functions of its child value nodes (note that a decision node cannot have another decision node as child in a simultaneous ID). In this sense, a node, if it is a child of a decision node, serves as an interface through which the choices of decision nodes may affect the value of the ID. Such a node is called an interface node. All interface nodes constitute an interface set. Collectively, an interface set serves as an interface of an ID through which policies can affect the expected value of the ID. Consequently, an ID can be divided into two fractions: the upstream fraction, which includes the interface nodes and the nodes "preceding" them, and the downstream fraction, which includes the interface nodes and the nodes "succeeding" the interface nodes.

Example: We use the ID in Fig. 1 to informally illustrate these concepts. The ID has two decision nodes $\{X_1, X_2\}$, five random nodes $\{A, B, C, D, H\}$, and one value node U. The interface set $\mathcal{Y}_{\text{in}}$ is $\{A, C\}$ since they have parental decision nodes. The upstream is $\{X_1, X_2, A, B, C\}$, which consists of two interface nodes A and C, node X_1 preceding node A, and nodes B and X_2 preceding node C. The downstream is

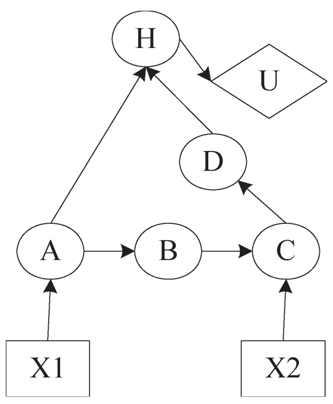


Fig. 1. ID to illustrate the representation theorem.

$\{A, C, H, D, U\}$, which consists of interface nodes A and C, the nodes H, D, U succeeding to the interface nodes. ■

Interestingly, corresponding to the structural separation that an ID can be divided into two fractions, the expected value of a value node under a policy breaks into two fractions, each of which involving only the upstream or the downstream of the ID.

B. *The Theorem*

We formalize the above idea in this section. For the sake of simplicity, throughout the paper, unless explicitly stated, we assume that: 1) the ID has only one value node; and 2) the value node has no decision node as its parent. We also note that our results in this paper generalize to the IDs with multiple value nodes and with value nodes having parental decision nodes. We relax these assumptions at the end of this section.

We begin by defining several concepts. A random node Y is an interface node if its parent set has at least one decision variable, i.e., $\pi(Y) \cap \chi \neq \emptyset$. The interface set of an ID is the set of all interface nodes. Due to the above assumptions, the interface set contains only random nodes; for this reason, we denote the set by $\mathcal{Y}_{\text{in}}$. The upstream of the ID includes the interface set and all ancestors of the nodes in the interface. By this definition, in addition to the interface random nodes and decision nodes, the upstream may contain the random-node ancestors of the interface nodes. These ancestral nodes must be included because they, together with decision nodes, determine the CPTs of the interface nodes. These ancestral random nodes form a set denoted by $\mathcal{Y}_0$.

Given an ID, we can efficiently identify its upstream using a queuing mechanism. We initialize a queue to be the interface set $\mathcal{Y}_{\text{in}}$ (it can be readily built by checking whether there is a parental decision node for every node in the ID) and the upstream $\mathcal{I}_{\text{up}}$ to be empty. At each step, a node is removed from the queue and added to $\mathcal{I}_{\text{up}}$ if it is not in $\mathcal{I}_{\text{up}}$. The parents of the node, if not present in $\mathcal{I}_{\text{up}}$ thus far, are added to the queue. The procedure terminates when the queue is empty. When it terminates, the set $\mathcal{I}_{\text{up}}$ becomes the upstream set. The procedure must terminate after a finite number of steps because an ID is a directed acyclic graph.

The upstream can be partitioned into three sets: the set $\mathcal{X}$ of decision nodes, interface set $\mathcal{Y}_{\text{in}}$, and the set $\mathcal{Y}_0$ of random-node ancestors of interface nodes. Given a policy δ, we define a function f_δ from $\Omega_{\mathcal{Y}_{\text{in}}}$ to the real line R. For notations, we let m be the number of nodes in the set $\mathcal{Y}_{\text{in}}$, $Y_{\text{in}}^{1:m}$ be a short notation of $\{Y_{\text{in}}^1, \ldots, Y_{\text{in}}^m\}$, and $y_{\text{in}}^{1:m}$ be an assignment to all interface variables, i.e., an element of $\Omega_{Y_{\text{in}}^{1:m}}$

$$f_\delta\left(y_{\text{in}}^{1:m}\right) = \sum_{Y\in\mathcal{Y}_0} \Pi_{Y\in\mathcal{Y}_0\cup\mathcal{Y}_{\text{in}}} P_\delta\left(Y|\pi(Y)\right) \Pi_{i=1}^n P_\delta(X_i) \quad (4)$$

where $\Pi_{Y\in\mathcal{Y}_0\cup\mathcal{Y}_{\text{in}}} P_\delta(Y|\pi(Y)) \Pi_{i=1}^n P_\delta(X_i)$ is the joint probability distribution of the variables in $\mathcal{X}$, $\mathcal{Y}_0$, and $\mathcal{Y}_{\text{in}}$, given policy δ. Hence, $f_\delta(Y_{\text{in}}^{1:m})$ is the conditional probability that the interface $Y_{\text{in}}^{1:m} = y_{\text{in}}^{1:m}$ occurs upon the policy δ. For convenience, we call them interface probabilities.

In contrast to the upstream, the downstream of an ID is the set consisting of all the interface nodes and their descendants. The downstream contains the value node, the interface nodes, and the random nodes that do not belong to the upstream. We use $\mathcal{Y}_1$ to denote the set of noninterface random nodes in the downstream. Note that the random nodes in the interface set belong to both the upstream and the downstream. For an assignment $y_{\text{in}}^{1:m}$ of the set $Y_{\text{in}}^{1:m}$ and the value node U, we can define a function as follows;

$$f_{\text{in},U}\left(y_{\text{in}}^{1:m}\right) = \sum_{Y\in\mathcal{Y}_1} \Pi_{Y\in\mathcal{Y}_1} P_\delta\left(Y|\pi(Y)\right) g_U\left(\pi(U)\right). \quad (5)$$

To see that $f_{\text{in},U}$ is a function of $Y_{\text{in}}^{1:m}$, we note that the interface variables may appear in $\pi(Y)$ for $Y \in \mathcal{Y}_1$. Given an assignment $y_{\text{in}}^{1:m}$ of $Y_{\text{in}}^{1:m}$, $f_{\text{in},U}(y_{\text{in}}^{1:m})$ is the expected utility conditioned on the assigned interface $y_{\text{in}}^{1:m}$. These quantities are called interface utilities for convenience. Since $\pi(Y)$ for Y in $\mathcal{Y}_1$ must belong to the downstream, $P_\delta(Y|\pi(Y))$ is independent of policy δ. Consequently, these utilities are independent of policy δ.

Theorem 1: Given a policy δ and a value node U, the expected value of the node U under policy δ

$$E_\delta[U] = \sum_{y_{\text{in}}^{1:m}\in\Omega_{Y_{\text{in}}^{1:m}}} f_\delta\left(y_{\text{in}}^{1:m}\right) \cdot f_{\text{in},U}\left(y_{\text{in}}^{1:m}\right). \quad (6)$$

Proof: We show that $E_\delta[U]$ can be rewritten as the sum of the interface utility $f_{\text{in},U}$ weighted by the probability f_δ over all interfaces

$$\begin{aligned} E_\delta[U] &= \sum_{\pi(U)} P_\delta\left(\pi(U)\right) g_U\left(\pi(U)\right) && \text{(a)} \\ &= \sum_{\pi(U)} \left[\sum_{\frac{\mathcal{Y}}{\pi(U)}} \Pi_{Y\in\mathcal{Y}} P\left(Y|\pi(Y)\right) \Pi_{i=1}^n P_\delta(X_i) \right] \\ &\quad \times g_U\left(\pi(U)\right) && \text{(b)} \\ &= \sum_{Y\in\mathcal{Y}_{\text{in}}} \sum_{Y\in\mathcal{Y}_0} \sum_{Y\in\mathcal{Y}_1} \Pi_{Y\in\mathcal{Y}_0\cup\mathcal{Y}_{\text{in}}} P\left(Y|\pi(Y)\right) \\ &\quad \times \Pi_{Y\in\mathcal{Y}_1} P\left(Y|\pi(Y)\right) \Pi_{i=1}^n P_\delta(X_i) g_U\left(\pi(U)\right) && \text{(c)} \end{aligned}$$

$$= \sum_{Y \in \mathcal{Y}_{\text{in}}} \left[\sum_{Y \in \mathcal{Y}_0} \Pi_{Y \in \mathcal{Y}_0 \cup \mathcal{Y}_{\text{in}}} P\left(Y \pi(Y)\right) \right]$$

$$\times \left[\sum_{Y \in \mathcal{Y}_1} \Pi_{Y \in \mathcal{Y}_1} P\left(Y | \pi(Y)\right) g_U\left(\pi(U)\right) \right] \quad \text{(d)}$$

$$= \sum_{y_{\text{in}}^{1:m} \in \Omega_{Y_{\text{in}}^{1:m}}} f_\delta\left(y_{\text{in}}^{1:m}\right) \cdot f_{\text{in},U}\left(y_{\text{in}}^{1:m}\right). \quad \text{(e)}$$

Step (a) is true by (2). At step (b), $\mathcal{Y}/\pi(U)$ is the difference set of $\mathcal{Y}$ and $\pi(U)$. This step is true by inserting (1) into (2). Step (c) follows from the fact that $\{\pi(U), \mathcal{Y}/\pi(U)\}$ and $\{\mathcal{Y}_0, \mathcal{Y}_{\text{in}}, \mathcal{Y}_1\}$ are two partitions of the set $\mathcal{Y}$. At step (d), we break the distribution $\Pi_{Y\in\mathcal{Y}} P(Y|\pi(Y)) \times \Pi_{i=1}^{n} P_\delta(X_i)$ into two fractions $\Pi_{Y\in\mathcal{Y}_0\cup\mathcal{Y}_{\text{in}}} P(Y|\pi(Y))$ and $\Pi_{Y\in\mathcal{Y}_1} P(Y|\pi(Y)) g_U(\pi(U))$. At step (e), we replace the two fractions with the definitions of the interface utilities and interface probabilities. ■

By the theorem, given a policy δ and a value node U, the expected value of the node under the policy can be represented as the sum of the multiplications of the interface utilities and corresponding interface probabilities.

Example (Continued): For the ID in Fig. 1, we show how to represent $E_\delta[U]$ for the value node U and a given policy δ. Let the policy δ be (δ_1, δ_2), where δ_i is the decision choice of X_i for $i = 1, 2$. By definition

$$E_\delta[U] = \sum_{H} \sum_{ABCD} P(A|\delta_1\delta_2) P(B|A) P(C|B\delta_2) \times P(D|C) P(H|AD) \Pi_{i=1}^{2} P_\delta(X_i) g_U(H).$$

The two functions are defined as follows.

$$f_\delta(A, C) = \sum_{B} P(A|\delta_1\delta_2) P(B|A) P(C|B\delta_2) \Pi_{i=1}^{2} P_\delta(X_i)$$

$$f_{\text{in},U}(A, C) = \sum_{HD} P(H|AD) P(D|C) g_U(H).$$

It can be verified that $E_\delta[U] = \sum_{A,C} f_\delta(A,C) \cdot f_{\text{in},U}(A,C)$. ■

We examine the assumptions we made at the beginning of this section. First, we have assumed that there is only one value node. In case of multiple value nodes, we may apply the representation theorem to each node. The expected value of a policy is the additive sum of the expected values of all value nodes under the policy.

Second, we have assumed that the value node has no decision nodes as its parents. In the other case that the value node has a decision node as its parents, the functions $f_{\delta,U}$ and f_{in} can be defined as follows, such that the theorem holds

$$f_{\delta,U}\left(Y_{\text{in}}^{1:m}\right) = \sum_{Y\in\mathcal{Y}_0} \Pi_{Y\in\mathcal{Y}_0\cup\mathcal{Y}_{\text{in}}} P_\delta\left(Y|\pi(Y)\right) \times \Pi_{i=1}^{n} P_\delta(X_i) g_U\left(\pi(U)\right)$$

$$f_{\text{in}}\left(Y_{\text{in}}^{1:m}\right) = \sum_{Y\in\mathcal{Y}_1} \Pi_{Y\in\mathcal{Y}_1} P_\delta\left(Y|\pi(Y)\right).$$

TABLE I
FACTORIZATION APPROACH TO ID EVALUATION

```
1. Pre-compute the quantities $f_{in,U}(y_{in}^{1:m})$ for all $y_{in}^{1:m}$ in $\Omega_{Y_{in}^{1:m}}$
2. For each policy $\delta$
       2.1 compute $f_\delta(y_{in}^{1:m})$ for each assignment of $Y_{in}^{1:m}$
       2.2 compute $E_\delta[U]$ by Equation (6)
3. Return the policy that maximizes $E_\delta[U]$
```

Therefore, we can lift the assumption that the value node has no decision node as parents. In this case, U is also called an interface node, but it belongs to the upstream only. The reason is that, by definition, a value node cannot have children and therefore cannot produce impact on the downstream.[2] Note that f_δ changes to $f_{\delta,U}$, since the value node is considered in computing the quantities relevant to the upstream. Interestingly, it can be proven that $f_{\text{in}}(= 1.0)$ is a constant. To see why, let us assume that the size of $\mathcal{Y}_1$ be k. We enumerate the set $\mathcal{Y}_1$ as $\{Y_1^1, \ldots, Y_1^k\}$ such that a node's parents appear after the node in the set. In computing $f_{\text{in}}(Y_{\text{in}}^{1:m})$, we can sequentially sum out the variables in $\mathcal{Y}_1$ in the enumerated order. Ultimately, we have $f_{\text{in}} = 1.0$.

C. The Algorithm

By the representation theorem, the expected value of a policy is represented as the sum of interface utilities weighted by the corresponding interface probabilities. The interface utilities are independent of the individual policies, whereas the interface probabilities are dependent on the policies. Therefore, the interface utilities can be factored out, i.e., they can be calculated once and reused across all the policies.

This is the idea behind our factorization approach, which is described in Table I. The factored-out computations are calculated once at line 1. They are used for all policies at line 2.2. Note that the procedure generalizes to IDs with multiple value nodes.

D. Complexity Analysis

It is of interest to compare the approach and the generic brute-forced approach that evaluates a policy directly by combining (1) and (2). Let n be the number of decision nodes. Thus, the size of the policy space is 2^n. Let the complexity of evaluating one policy be C. The complexity C breaks into three pieces: computing f_δ, computing $f_{\text{in},U}$, and computing $E_\delta[U]$ by (6). We denote them, respectively, by C_1, C_3, and C_2. To evaluate all policies, the generic approach has complexity $2^n C$, i.e., $2^n(C_1 + C_2 + C_3)$. In contrast, since the factorization approach computes $f_{\text{in},U}$ only once but uses them 2^n times, its complexity is $2^n(C_1 + C_2) + C_3$. A good measure to predict the computational gain is the size of the downstream, i.e., the number of nodes in it. In one extreme, if the downstream contains only the interface nodes and the value nodes (thus, C_3 is a constant), the two approaches have the same complexity. In the other extreme, if the downstream contains far more nodes

[2]Note that this is different from random nodes having parental decision nodes. Such a random node belongs to both the upstream and the downstream.

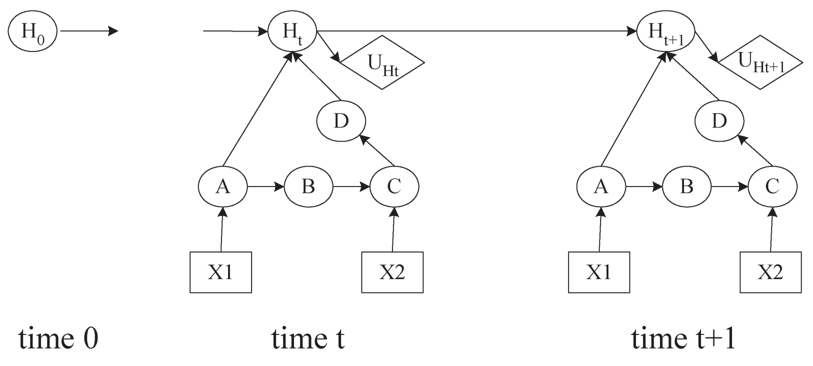


Fig. 2. Dynamic ID model.

than the upstream (i.e., $C_3 \gg C_1 + C_2$), the computational gain is significant.

E. *Bounding the Optimal Expected Value*

We show that the interface utilities computed in the factorization approach can be used to derive both an upper and a lower bound of the optimal value of the ID. These bounds have significant implications in practical planning.

We define $f^{+}_{\mathrm{in},U}$ and $f^{-}_{\mathrm{in},U}$ to be the largest and the smallest one among all interface utilities, i.e.,

$$f^{+}_{\mathrm{in},U} = \max_{y^{1:m}_{\mathrm{in}} \in \Omega_{Y^{1:m}_{\mathrm{in}}}} f_{\mathrm{in},U}\left(y^{1:m}_{\mathrm{in}}\right)$$

$$f^{-}_{\mathrm{in},U} = \min_{y^{1:m}_{\mathrm{in}} \in \Omega_{Y^{1:m}_{\mathrm{in}}}} f_{\mathrm{in},U}\left(y^{1:m}_{\mathrm{in}}\right)$$

where the $\max$ and $\min$ are taken over the domain of the variables in $Y^{1:m}_{\mathrm{in}}$. From the theorem, we see that $f^{+}_{\mathrm{in},U}$ $(f^{-}_{\mathrm{in},U})$ is the upper (lower) bound of the optimal value of the ID. These bounds have significant importance in practice. Suppose, for instance, that these bounds are available to a planner. In one extreme, if the planner expects a utility that is larger than the upper bound, he never bothers to evaluate all the policies and finds the optimal one because even the best policy provides less than he expects. In this case, he needs to redesign the network structure or parameters such that the performance of the ID can be improved. In the other extreme, if the planner expects a utility that is less than the lower bound, again he never bothers to evaluate all the policies and chooses the optimal one because any policy can provide more than he expects. In this case, he can pick any policy and execute it.

We note that from the computational point of view, computing these bounds is easier than evaluating the ID. There are two reasons. First, as discussed earlier, computing these bounds involves only the downstream of the ID, whereas evaluating an ID involves its entire structure. If the downstream contains much fewer variables than the upstream, the interface utilities (and also the bounds) can be obtained efficiently. Second, computing these bounds avoid enumerating all the policies and calculating their expected values.

Finally, the tightness of the bounds depends on the structure of an ID, the CPTs of random nodes, and the value functions of value nodes. It is difficult to characterize a general condition to determine the tightness of the bounds. In our experiments, we empirically show that these bounds are reasonably tight for the tested problems.

V. Extensions to the Factorization Approach

In this section, we discuss two extensions to the factorization approach. These extensions deal with reconstructing the policies as the network structure/parameters undergo changes. There are two perspectives. First, at one decision step, the network might change such as more actions being available for a planner's choice, more value nodes needed consideration, and so on. Second, the network might dynamically alter its structure or parameters as time goes by. For example, if a subgoal is successfully accomplished at one step, it can be removed from the network in the subsequent steps.

A. *Network Structure/Parameter Changes*

The principle for the factorization approach to accommodate structure or parameter changes is as follows. First, if the changes involve only the upstream of an ID, the interface utilities do not need to be recomputed and can still be shared in evaluating the ID. Specifically, these changes include addition or removal of decision/random/value nodes and also the alternation of CPTs and value functions in the upstream. Second, if the changes involve only the downstream of the ID, the approach needs to reconstruct the interface utilities. Fortunately, the interface probabilities are preserved and the calculations for them can be saved. Third, if the changes involve not only the upstream but also the downstream, the approach needs to recompute both the interface utilities and the interface probabilities.

B. *Planning Over Time*

In realistic applications, network parameters may change over time. In this case, we can use a dynamic ID to model the conditional dependencies among nodes over time. In this section, we show how the factorization approach can be used to reconstruct the policies on a step-by-step basis for dynamic IDs.

To facilitate our discussions, we extend the example in Fig. 1 to a dynamic ID. We assume that the variable H evolves over time and let H_t denote H at step t. The dynamic ID is drawn in Fig. 2. In contrast, the ID in Fig. 1 is said to be static since the multiple decisions are made at one time step.

The dynamic ID has two prominent features. First, at a single step, the decision problem can be modeled as a static ID. In addition to the nodes and links in Fig. 1, the node H_{t+1} at step $t+1$ has one more parent node H_t. Second, the intertemporal link between two consecutive nodes carries the historic information about the sequence of performed policies. For step $n+1$, the information can be summarized by a probability distribution of H_t conditioned on the history [14].

For a dynamic ID, we are interested in optimal planning on the step-by-step basis. The problem is formulated as: Given an initial probability distribution $P(H_0)$, at step $t+1$, how to efficiently find the policy $\delta[=(\delta_1,\ldots,\delta_n)$ where n is the number of decision nodes] that maximizes $E_\delta[U_{H_{t+1}}]$? To solve the problem, we show: 1) how to select the optimal policy at step $t+1$, given the probability distribution $P(H_t)$; and 2) how to sequentially update the probability distribution $P(H_{t+1})$ from $P(H_t)$, given a policy δ at the previous step. After these two questions are settled, we may choose the optimal policy as follows. At step $t+1$, we first choose the optimal policy for the step and then update the probability $P(H_{t+1})$ from $P(H_t)$. The procedure repeats at each step.

To answer the first question, we introduce the concept of an augmented interface node and an augmented interface set. We call the node H_t an augmented interface node of the ID at step $t+1$ since the node H_t can produce impact on the network via altering its probability distribution. In this sense, it is an interface node.[3] The augmented interface set consists of $\mathcal{Y}_{\text{in}}$ as before and the node H_t. The downstream of the ID at step $t+1$ remains the same as that of the static ID. Likewise, we may define the two functions f_δ and $f_{\text{in},U_{H_{t+1}}}$. Therefore, we can use the factorization approach to solve the planning problem over time. The computations involving $f_{\text{in},U_{H_{t+1}}}$ are factored out. Note that these interface utilities are shared for all policies at each decision step. For the ID in Fig. 2, we can define the following functions for the ID:

$$\begin{aligned}
&f_\delta(A,C,H_t)\\
&\quad = P(H_t)\sum_B P(A|\delta_1\delta_2)P(B|A)P(C|B\delta_2)\Pi_{i=1}^2 P_\delta(X_i)\\
&f_{\text{in},U_{H_t}}(A,C,H_t)\\
&\quad = \sum_{H_{t+1}D} P(H_{t+1}|ADH_t)P(D|C)g_U(H_{t+1}).
\end{aligned}$$

It can be verified that $E_\delta[U_{H_{t+1}}]=\sum_{A,C,H_t} f_\delta(A,C,H_t)\cdot f_{\text{in},U_{H_t}}(A,C,H_t)$.

To answer the second question, we show how to efficiently compute $P(H_{t+1})$, given a distribution $P(H_t)$ and the policy δ performed at step t. We introduce a technique such that the procedure of computing $P(H_{t+1})$ can be conducted similar to that of computing $E_\delta[U_{H_{t+1}}]$. Suppose that H_{t+1} can take on two values h(true) and $\neg h$(false). We first show how to calculate the probability of H_{t+1} being true. Let V be a value node that differs from $U_{H_{t+1}}$ only in its value function. Specifically, g_V is 1.0 if its parent H_{t+1} is true; it is 0.0 otherwise. For simplicity, let $H_{t+1}=h(\neg h)$ denote the event that the hypothesis H_{t+1} is true (false). We prove that $E_\delta[V]=P_\delta(H_{t+1}=h)$.

Proposition 1: $E_\delta[V]=P_\delta(H_{t+1}=h)$.

Proof:

$$\begin{aligned}
E_\delta[V] &= \sum_{H_{t+1}} P_\delta(H_{t+1})g_V(H_{t+1})\\
&= P_\delta(H_{t+1}=h)g_V(H_{t+1}=h)\\
&\quad + P_\delta(H_{t+1}=\neg h)g_V(H_{t+1}=\neg h)\\
&= P_\delta(H_{t+1}=h).
\end{aligned}$$

In the last step, we use the definition of the value function g_V.■

To calculate the probability of H_{t+1} being false, we may define g_V as follows: It is 1.0 if its parent H_{t+1} is false; it is 0.0 otherwise. If we define two functions f_δ and $f_{\text{in},V}$, we see that the computational steps for $E_\delta[V]$ are the same as those for computing $E_\delta[U_{H_{t+1}}]$. Hence, computing $P(H_{t+1})$ does not add much overhead to ID evaluation.

It is interesting to compare the generic approach and the factorization approach in the context of the dynamic ID. Let the number of decision steps be T. Recall that the complexity of computing f_δ is C_1, the complexity of computing $f_{\text{in},U_{H_{t+1}}}$ is C_3, and the complexity of computing $E_\delta[U_{H_{t+1}}]$ is C_2. Since C_1 takes constant time, it can be ignored. For one decision step, the factorization approach has the complexity $2^nC_2+C_3$ while the generic approach has the complexity $2^n(C_2+C_3)$. For T steps, the complexity of the factorization approach is $2^nTC_2+C_3$ [note that this does not include the overhead of computing $P(H_{t+1})$], while the complexity of the generic approach is $2^nT(C_2+C_3)$. If $C_3 \gg C_2$, the factorization approach can be extremely efficient.

VI. Experiments

In this section, we report our experiments on both simulation studies and a military planning example. In our experiments, we wrote Matlab-V6.5 codes and ran them on a laptop with a 2.0-GHz central processing unit (CPU) under Windows XP. We compare the factorization approach against the generic brute-forced approach. We chose the generic approach because we were not aware of specific algorithms for evaluating simultaneous IDs. For convenience, we refer to the two algorithms as `evalCS` (named after computation sharing) and `evalBF` (named after brute forced).

A. Simulation Studies

To thoroughly evaluate the performance of the factorization approach, we conducted simulated studies on the ID in the left chart of Fig. 3, which is similar to the military planning examples in [15]. It is referred to as the static ID in the rest of this section. The CPTs are randomly generated. The value functions for value nodes are manually specified.

Specifically, our experiments are designed to: 1) evaluate the performances of the factorization approach for static and

[3]Previously, we defined an interface node to be a node that has parental decision nodes since the choices of decision nodes can affect its CPTs and in turn, the expected value of the ID. In contrast, the node H_t is called an augmented interface node since it can change its probability distribution and thus, affect the expected value of the ID.

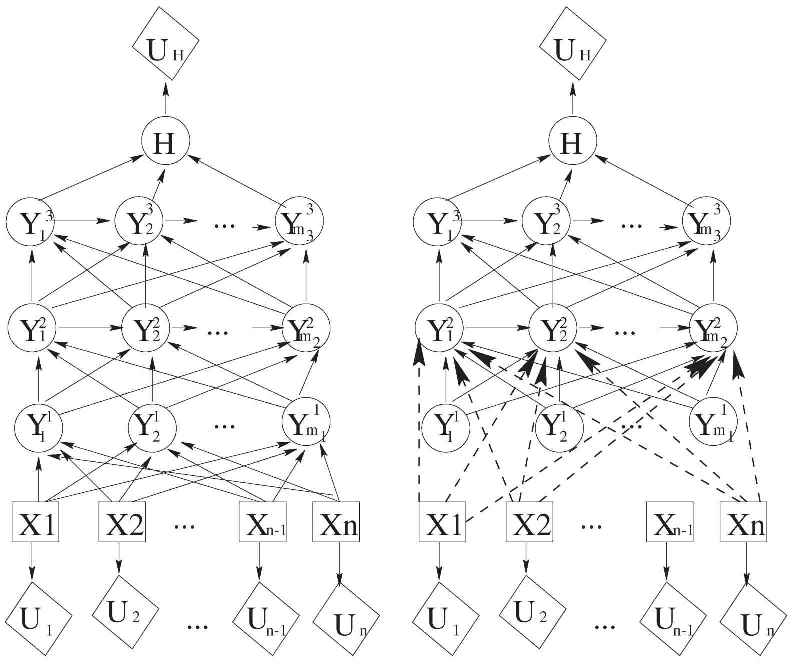


Fig. 3. Test example is shown in the left chart, while the right is its variant for comparative studies.

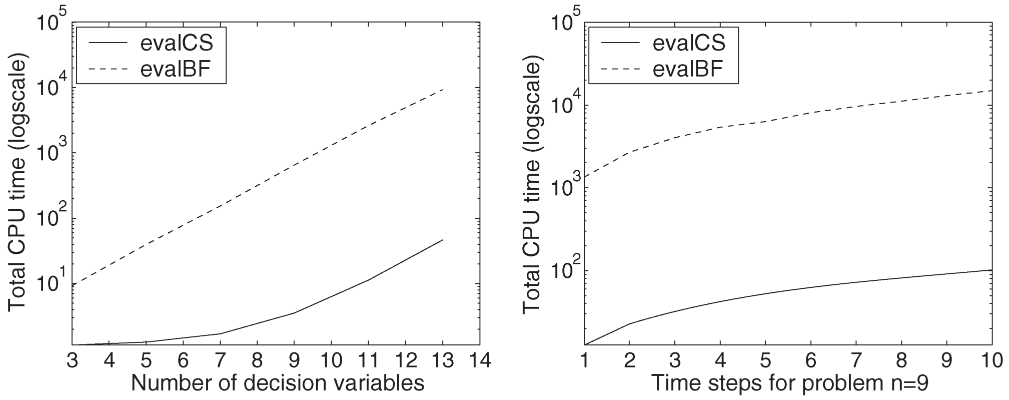


Fig. 4. Performance comparison of `evalBF` and `evalCS`.

dynamic IDs; 2) show the tightness of bounds derived from the interface utilities; 3) demonstrate how the computational gain achieved by the approach varies with different network structures; and 4) demonstrate the computational gain by adapting the approach to account for newly added decisions and value nodes.

1) Performances of the Factorization Approach: To see how the performances of the algorithms vary with the number of decision nodes, we fix the number of random nodes at each level at four and vary the number of decision nodes. Thus, the static ID with n decision nodes has additionally ten random nodes and $n+1$ value nodes. We ran `evalBF` and `evalCS` for seven problems with $n = 3, 5, \ldots, 13$. The timing data are presented in the left chart of Fig. 4. The chart gives the total CPU seconds that the algorithms took for each of the problems. Note that the vertical direction is drawn in log scale. The solid (dashed) curve is for `evalCS` (`evalBF`). It can be seen that `evalCS` is considerably more efficient than `evalBF`. For instance, from our data, for $n = 9$, to evaluate 512 policies, `evalCS` took 3.51 s while `evalBF`, 646.74 s; for $n = 13$, to evaluate 8192 policies, `evalCS` used 46.32 s while `evalBF`, 9284.05 s.

To quantitatively characterize how much savings the factorization procedure can bring about, we use the timing results of `evalCS` to predict the performance of `evalBF`. Recall that the complexity of evaluating a policy breaks into three fractions C_1, C_3, and C_2. We ignore C_1 since it is a constant. For each problem, we estimate C_3 by the actual seconds $\hat{C}_3$ of computing f_{in,U_H}, and C_2 by $\hat{C}_2$ as $(\text{the total CPU time} - \hat{C}_3)/(\text{the number of policies})$. The complexity of `evalBF` is predicted by $2^n(\hat{C}_2 + \hat{C}_3)$. We found that these estimations are almost the same as the actual timing results of `evalBF`. This suggests the effectiveness of our complexity analysis.

We also tested the algorithms over the dynamic ID in Fig. 5, which is an extension of the left chart of Fig. 3.

Initially, the probability of the node H being false is set to 1.0. Its probability is updated at each decision step. We ran both algorithms for up to ten decision steps. We showed the

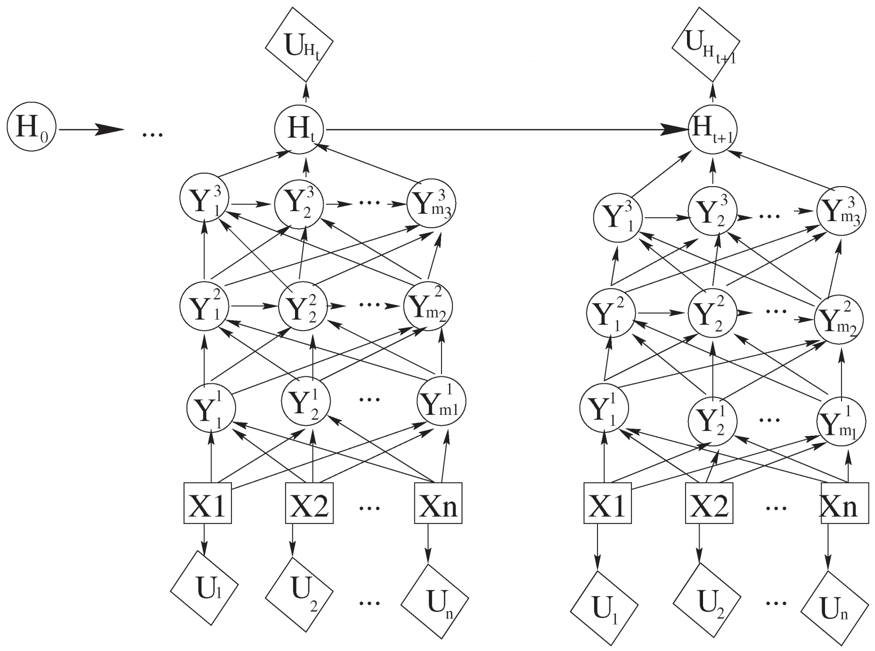


Fig. 5. Dynamic influence diagram.

total CPU time for the ID with nine decision nodes in the right chart of Fig. 4. The chart gives the total CPU seconds for both algorithms against the time steps. Note that again the vertical direction is drawn in log scale. It can be seen that the CPU time linearly increases with the elapsed time for `evalBF` while its increase is negligible for `evalCS`. This is not a surprising observation. In `evalBF`, all policies are evaluated at each step. The time cost for all steps remains the same. Hence, the increase is linear. From our data, `evalBF` uses about 1300 s to evaluate all 512 (2^9) policies at each step. However, in `evalCS`, the interface utilities are computed only once at the first step. So, we observe that at the first step, `evalCS` takes about 2.66 s to compute these utilities; thereafter, each step takes about only 10 s to evaluate all 512 policies. The increase is negligible when compared against that in `evalBF`.

2) Tightness of Bounds: To show the tightness of the upper and lower bounds of the optimal expected value, in Fig. 6, we plot the optimal value (the middle curve) and these bounds (the upper and lower curves) for the static IDs with 3, 5, ...,13 decision nodes. We see that these bounds are reasonably tight for the tested problems. For example, for $n = 7$, the optimal value is 871.47 while the bounds are 722.91 and 936.637. Although it is difficult to quantitatively analyze the properties of these bounds, these experiments show they can be tight at least for these tested problems.

3) Computational Gain Under Network-Structure Changes: To demonstrate how the computational gain of `evalCS` varies with different network structures, we run `evalCS` over the static ID and a modified version of it. The modified ID is obtained as follows: Every link from X_i to Y_j^1 is redirected to Y_j^2. The resulting ID is shown in the right chart of Fig. 3. Its upstream is $\mathcal{X} \cup Y^1_{1:m_1} \cup Y^2_{1:m_2} \cup U_{1:n}$, whereas its downstream is $Y^3_{1:m_3} \cup \{H\} \cup \{U_H\}$, where $U_{1:n}$ means the set of value nodes, and $Y^i_{1:m_i}$ means the set of nodes Y^i_j, i.e., $Y^i_{1:m_i} = \{Y^i_1, \ldots, Y^i_{m_i}\}$ for $i = 1, 2, 3$. Compared with that of

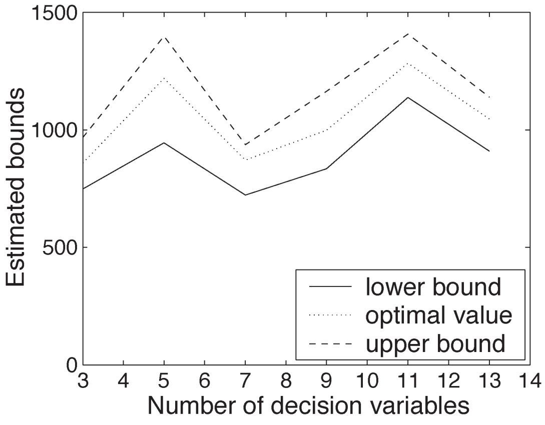


Fig. 6. Lower and upper bounds obtained from the interface utilities.

the static ID, the downstream of the modified ID contains fewer random nodes. We expect: 1) `evalCS` is still more efficient than `evalBF` in the modified ID, since its downstream contains a number of random nodes; and 2) `evalCS` achieves less savings in modified ID than it does in the original ID.

The experiments presented in Fig. 7 confirm these expectations. First, the left chart plots the CPU seconds (in log scale) that `evalCS` and `evalBF` take for the modified IDs with a different number of decision variables. It can be seen that `evalCS` is more efficient. Second, the right chart plots the magnitudes of the savings brought by `evalCS`. For each approach, the saving magnitude is measured by the quotient of the total time of `evalBF` and that of `evalCS`. The magnitudes are drawn in the vertical direction. For a modified ID, `evalCS` is about 14 times faster than `evalBF`. For the original IDs, the magnitudes vary with the number of their decision nodes. We see that the computational savings brought by `evalCS` are more significant for IDs whose downstream contains more nodes.

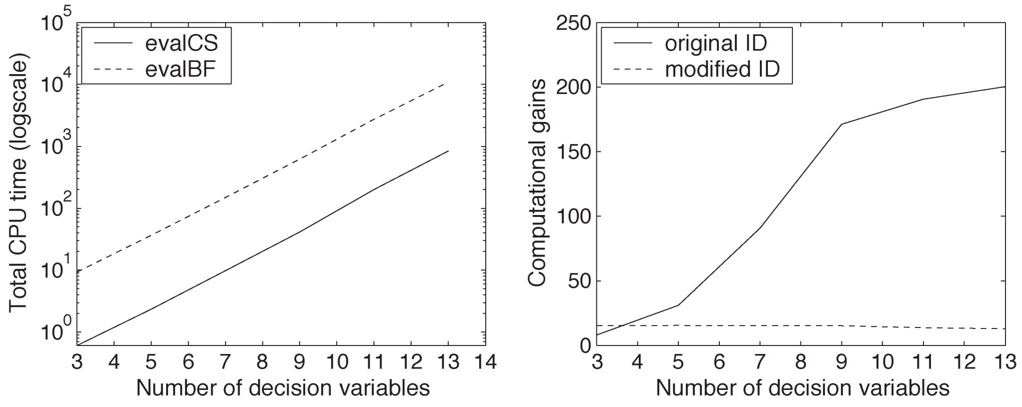


Fig. 7. Computational gains versus network structure.

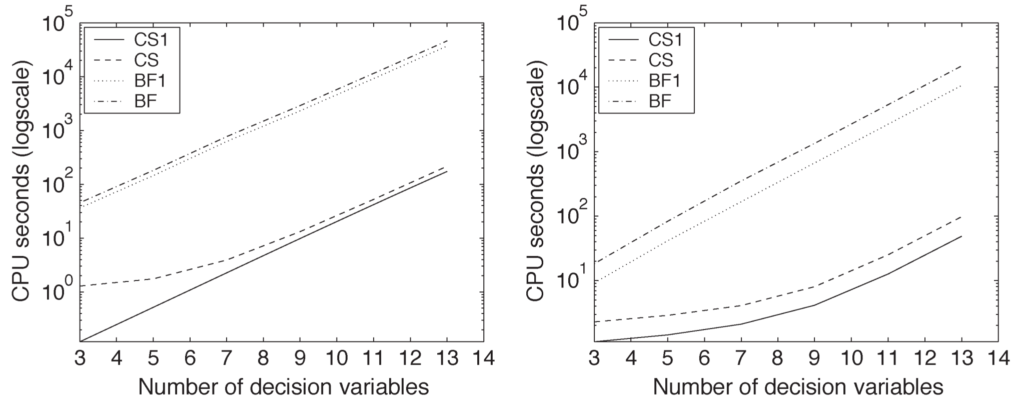


Fig. 8. Replanning for added action/value nodes. In both charts, the curves from the top and bottom plot the total/replanning time for the generic approach, and the total/replanning time for the factorization approach.

4) Computational Gain Under Network-Parameter Changes: We also conducted experiments to show how the factorization approach achieves computational savings as the network changes. For this purpose, given the static ID, we first evaluate it (the planning phase), then add more nodes to the ID and reevaluate it (the replanning phase). We like to compare both the replanning time and total time of the factorization approach against that of the generic approach.

In one experiment, we first evaluate a static ID with n decision nodes. We then add two decision nodes to the ID and evaluate the modified ID. Every newly added node has a link from itself to every Y_j^1 node. The timing results in log scale are presented in the left chart of Fig. 8. In the chart, the curve corresponding to `CS1` (`BF1`) depicts the replanning time for the factorization (generic) approach, whereas the curve corresponding to `CS` (`BF`) depicts the total time similarly. We see that for the tested problems, the factorization approach achieves considerable savings in replanning when more decision nodes are added. For instance, for the ID with nine decision nodes, the factorization and generic approach, respectively, takes 9.80 and 2313.94 s. These savings are achieved through sharing the interface utilities computed during the evaluation of the original ID. Since the factorization approach takes much less time in both evaluating and reevaluating the ID, its total time is considerably less than that used by the generic approach.

In the other experiment, we evaluate the ID and then add one more value node for replanning. The added value node has a link from every node Y_j^2 for $j = 1, \ldots, 4$. In computing the expected value of the added value node, we still use the factorization approach. In reevaluating an ID, we do not recompute the expected value of the existing value node. The timing results in log scale are collected in the right chart of Fig. 8. The legends read similar to those in the left chart. It can be seen that the factorization approach can achieve great savings in replanning. The reason is obvious: The factorized computations are saved in computing the expected value of the newly added value node. By taking advantage of shared computations in evaluating two value nodes, the total time used by the factorization approach is considerably less than that by the generic approach.

B. A Military Planning Example

We applied the factorization approach to a hypothetical military planning example, which is illustrated in Fig. 9. The overall military goal is to win a war or to bring a tyrant to justice. The goal is represented by a `Hypothesis` node, which is on the top of the figure. There are 12 primitive actions, namely `destroy_C2`, `destroy_Radars`, ..., `operate_special_force`, which are on the bottom side. Performing an action has direct effects of specific purpose. For instance, if the action `destroy_Radars` is performed, the `EW/GCIRadars` is destroyed with a high probability. These effects alter the overall goal through altering the low-level subgoals. For instance, the status

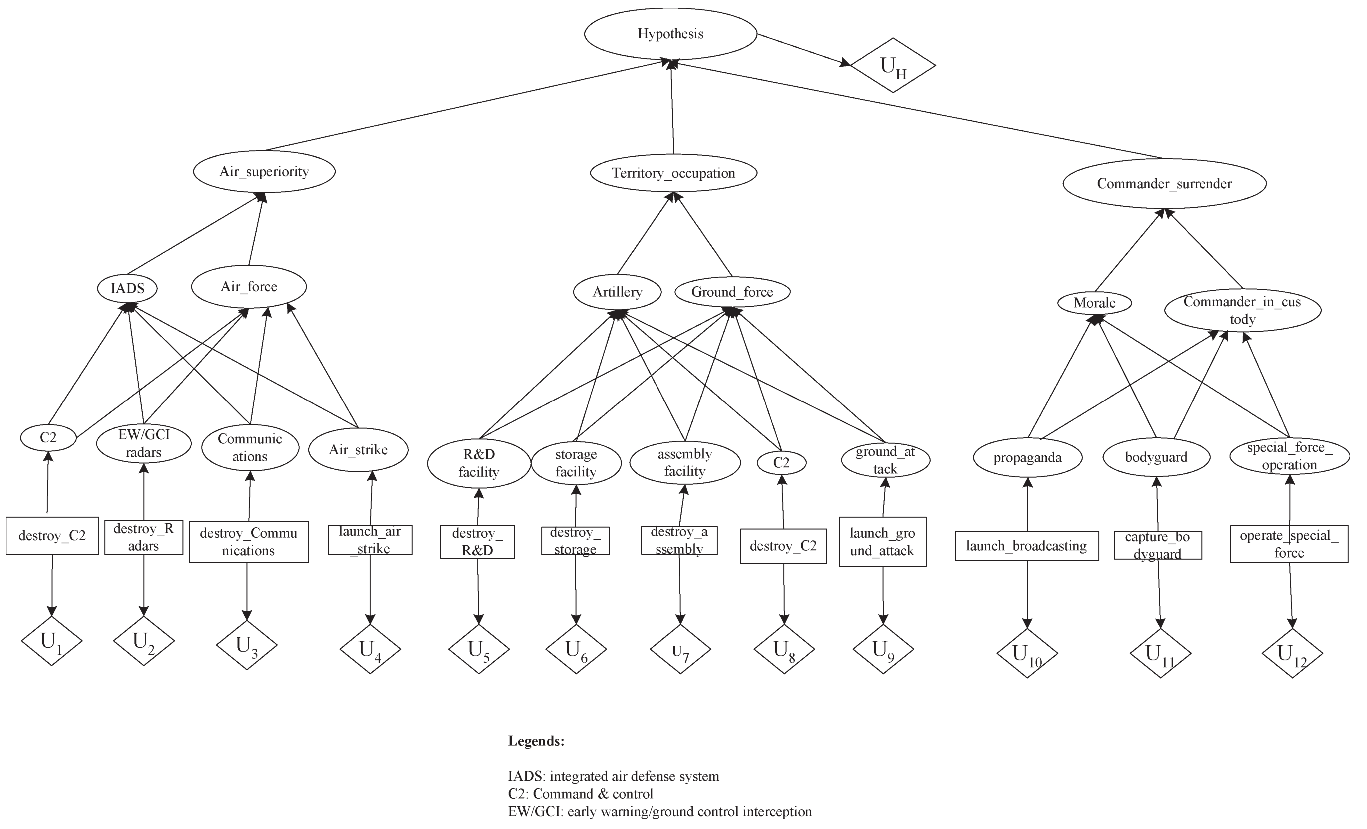


Fig. 9. Static ID illustrating a military planning problem.

of `C2` (command and control), `EW/GCIRadars` and `Communications` facilities, and `Air_strike` determine the workability of the integrated air defense system (IADS) and the strength of the enemy air force. In turn, the workability of IADS system and the strength of the enemy air force determine the loss of `Air_superiority`. The `Air_superiority`, `Territory_occupation`, and `Commander_surrender` are three subgoals determining the overall goal success. Without loss of generality, we assume all nodes are binary. In the example, each decision node is associated with a value node encoding the cost of performing the action, and the hypothesis node is associated with a value node encoding the utility of goal success. The optimal policy needs to balance the utility of goal success and cost of performing actions.

We designed a reasonable set of CPTs and value functions. For the `Hypothesis` node, if all the subgoals `Air_superiority`, `Territory_occupation`, and `Command_surrender` are achieved, the overall goal is successfully achieved. If one of the subgoals is to be achieved, the probability of the overall success is decreased by 0.3; however, if none of the subgoals is achieved, the overall goal fails with certainty. Similarly, for the subgoal `Air_superiority`, the two influencing factors are `IADS` and `Air_force`. If the IADS system works well and `Air_force` is strong, `Air_superiority` is true for the enemy air force; if either the IADS system works poorly or `Air_force` is weak, the probability of `Air_superiority` being true is decreased by 0.5. Other CPTs for `Territory_occupation` and `Command_surrender` are set analogously to those for `Air_superiority`. A similar strategy is used in parameterizing the nodes `IADS`, `Air_force`, `Artillery`, `Ground_force`, `Morale`, and `Commander_in_custody`. In determining the CPTs for random nodes that are immediate children of the decision nodes, we assume that an action achieves its intended effect with probability 0.9. For example, a `destroy_Radars` decision will destroy the EW/GCI radars with probability 0.9. To complete the ID definition, we also assigned value functions. If the goal is successfully achieved, the reward is 1000; otherwise, the cost, i.e., a negative reward, is 500. For other decision nodes, if a ground attack is launched, the cost is 150; if the special force operation is performed, its cost is 100; if the commander decides to capture the bodyguards of the tyrant, the operating cost is 80; if an air strike is launched, the cost is 50; for any other actions, their operating cost is 20.

Our primary interest is in the performance of the factorization algorithm. From our data, to evaluate the ID, the factorization algorithm took 45 s, while the brute-forced algorithm took 9012 s. Hence, the computational saving is tremendous. We can explain the performance difference by the ID structure—its downstream contains a large number of nodes: all random nodes and the value node associated with the goal. Since its downstream contains far more random nodes than its upstream, the approach is expected to be significantly more efficient. Our secondary interest is concerned with the optimal policy. The optimal policy is the one that performs only air strike and special force operation. The expected value of the ID is 561.98, and the probability of goal success is 0.81. We note that the optimal policy excludes "launching a ground attack," although it is the action that is most likely to lead to goal success. One possible reason, as explained earlier, is that the action is excluded due to the high operating cost of performing the action.

VII. Conclusion and Future Work

In this paper, we studied a special ID class, namely simultaneous IDs, where multiple decisions need to be made at one time step. We intended to make two contributions. First, we examined a simultaneous ID and studied its theoretical properties. We showed that such an ID can be decomposed into an upstream fraction and a downstream fraction, and that the expected value of a value node under a policy can be represented as the sum of interface utilities that involve only the downstream fraction, weighted by the corresponding interface probabilities that involve only the upstream fraction. The interface utilities naturally provide an upper and lower bound of the optimal value of the ID. Second, we proposed a novel factorization algorithm to evaluate a simultaneous ID. The interface utilities are independent of the individual policies; therefore, they can be calculated once but used across all policies in evaluating them. We also extend the factorization approach to a dynamic ID. The algorithm has been tested on simulation studies and a military planning example. Our experiments showed that the factorization algorithm is significantly more efficient than the generic algorithm in evaluating a simultaneous ID.

To further speed up ID evaluating, one future direction is to combine the factorization approach with the approaches of reducing the search space. In this paper, we address one difficulty in ID evaluation, i.e., evaluating individual policies. Another difficulty in ID evaluation is that the policy space contains exponentially many polices and one needs to evaluate all of them in order to find the optimal one. The ID evaluation process can be accelerated if the technique in this paper can be integrated with the approaches of reducing the search space.

Acknowledgment

The authors would like to thank J. Lemmer, D. Gossink, and J. Dussault from AFRL/Rome for introducing the problem to us and for numerous technical exchanges on the related issues. The authors are grateful to three anonymous reviewers for their insightful comments in improving the paper. The authors are also grateful to W. Liao and Z. Zhu for insightful discussions.

References

[1] R. Howard and J. Matheson, "Influence diagrams," in *The Principles and Applications of Decision Analysis*, R. Howard and J. Matheson, Eds. Menlo Park, CA: Strategic Decisions Group, 1984, pp. 719–762.
[2] H. Raiffa, *Decision Analysis*. Reading, MA: Addison-Wesley, 1968.
[3] R. D. Shachter, "Evaluating influence diagrams," *Oper. Res.*, vol. 34, no. 6, pp. 871–882, 1986.
[4] P. P. Shenoy, "Valuation-based systems for Bayesian decision analysis," *Oper. Res.*, vol. 40, no. 3, pp. 463–484, 1992.
[5] F. Jensen, F. V. Jensen, and S. L. Dittmer, "From influence diagram to junction trees," in *Proc. 10th Conf. Uncertainty Artificial Intelligence*, Seattle, WA, 1994, pp. 367–373.
[6] R. Qi and D. Poole, "A new method for influence diagram evaluation," *Comput. Intell.*, vol. 11, no. 1, pp. 1–34, 1995.
[7] N. L. Zhang and D. Poole, "Stepwise-decomposable influence diagram," in *Proc. 3rd Int. Conf. Principles Knowledge Representation and Reasoning*, Cambridge, MA, 1992, pp. 141–152.
[8] G. F. Cooper, "A method for using belief networks as influence diagrams," in *Proc. 4th Workshop Uncertainty Artificial Intelligence*, St. Paul, MN, 1988, pp. 55–63.
[9] R. Shachter and M. Peot, "Decision making using probabilistic inference methods," in *Proc. 8th Annu. Conf. Uncertainty Artificial Intelligence (UAI)*, Stanford, CA. San Mateo, CA: Morgan Kaufmann, 1992, pp. 276–283.
[10] T. Nielsen and F. Jensen, "Well-defined decision scenarios," in *Proc. 15th Conf. Uncertainty Artificial Intelligence*, Stockholm, Sweden, 1999, pp. 502–511.
[11] F. Jensen and M. Vomlelova, "Unconstrained influence diagrams," in *Proc. 18th Conf. Uncertainty Artificial Intelligence*, Edmonton, AB, Canada, 2002, pp. 234–241.
[12] S. L. Lauritzen and D. Nilsson, "Representing and solving decision problems with limited information," *Manage. Sci.*, vol. 47, no. 9, pp. 1238–1251, 2001.
[13] N. L. Zhang, "Probabilistic inferences in influence diagrams," in *Proc. 14th Conf. Uncertainty Artificial Intelligence*, Madison, WI, 1998, pp. 514–522.
[14] K. J. Aström, "Optimal control of Markov decision processes with incomplete state estimation," *J. Math. Anal. Appl.*, vol. 10, no. 3, pp. 174–205, 1965.
[15] U. Kuter, D. Nau, and J. F. Lemmer, "Interactive planning under uncertainty with causal modeling and analysis," Dept. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. CS-TR-4434, 2003.

Weihong Zhang received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology, Kowloon, Hong Kong, in 2001.

He then worked as a Postdoc Researcher with the Department of Computer Science and Engineering, Washington University, Saint Louis, MO. He is currently a Postdoc Researcher with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. He has conducted research in artificial intelligence, probabilistic inferences and decision making under uncertainty, graphical models and their applications in sensor networks and human–computer interaction. He has published more than 10 papers in peer-reviewed journals and conferences.

Qiang Ji (S'92–M'98–SM'04) received the Ph.D. degree in electrical engineering from the University of Washington in 1998.

He is currently an Associate Professor with the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY. His areas of research include computer vision, probabilistic reasoning for decision making and information fusion, pattern recognition, and robotics. He has published more than 70 papers in peer-reviewed journals and conferences. His research has been funded by local and federal government agencies including National Science Foundation (NSF), National Institutes of Health (NIH), Air Force Office of Scientific Research (AFOSR), Office of Naval Research (ONR), Defense Advanced Research Projects Agency (DARPA), and Army Research Office (ARO) and by private companies including Boeing and Honda. His latest research focuses on face detection and recognition, facial-expression analysis, image segmentation, object tracking, user affect modeling and recognition, and active information fusion for decision making under uncertainty.

International Journal of Approximate Reasoning 49 (2008) 436–450

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Efficient non-myopic value-of-information computation for influence diagrams

Wenhui Liao [a,*], Qiang Ji [b]

[a] Research & Development, Thomson-Reuters Corporation, 610 Opperman Drive, Eagan, MN 55123, USA
[b] Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA

ARTICLE INFO

Article history:
Received 4 January 2007
Received in revised form 17 April 2008
Accepted 21 April 2008
Available online 29 April 2008

Keywords:
Value-of-information
Influence diagrams
Decision making
Central-limit theorem
Stress modeling

ABSTRACT

In an influence diagram (ID), value-of-information (VOI) is defined as the difference between the maximum expected utilities with and without knowing the outcome of an uncertainty variable prior to making a decision. It is widely used as a sensitivity analysis technique to rate the usefulness of various information sources, and to decide whether pieces of evidence are worth acquisition before actually using them. However, due to the exponential time complexity of exactly computing VOI of multiple information sources, decision analysts and expert-system designers focus on the myopic VOI, which assumes observing only one information source, even though several information sources are available. In this paper, we present an approximate algorithm to compute non-myopic VOI efficiently by utilizing the central-limit theorem. The proposed method overcomes several limitations in the existing work. In addition, a partitioning procedure based on the *d*-separation concept is proposed to further improve the computational complexity of the proposed algorithm. Both the experiments with synthetic data and the experiments with real data from a real-world application demonstrate that the proposed algorithm can approximate the true non-myopic VOI well even with a small number of observations. The accuracy and efficiency of the algorithm makes it feasible in various applications where efficiently evaluating a large amount of information sources is necessary.

1. Introduction

In a wide range of decision-making problems, a common scenario is that a decision maker must decide whether some information is worth collecting, and what information should be acquired first given several information sources available. Each set of information sources is usually evaluated by value-of-information (VOI). VOI is a quantitative measure of the value of knowing the outcome of the information source(s) prior to making a decision. In other words, it is quantified as the difference in value achievable with or without knowing the information sources in a decision-making problem.

Generally, VOI analysis is one of the most useful sensitivity analysis techniques for decision analysis [23,25]. VOI analysis evaluates the benefit of collecting additional information in a specific decision-making context [27]. General VOI analyses usually require three key elements: (1) A set of available actions and information collection strategies; (2) A model connecting the actions and the related uncertainty variables within the context of the decision; and (3) values for the decision outcomes. The methods of VOI analysis could be quite different when different models are used.

In this paper, we consider VOI analysis in decision problems modeled by influence diagrams. Influence diagrams were introduced by Howard and Matheson in 1981 [13] and have been widely used as a knowledge representation framework

* Corresponding author. Tel.: +1 5189610457.
E-mail addresses: wenhui.liao@thomsonreuters.com (W. Liao), jiq@rpi.edu (Q. Ji).

0888-613X/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2008.04.003

to facilitate decision making and probability inference under uncertainty. An ID uses a graphical representation to capture the three diverse sources of knowledge in decision making: conditional relationships about how events influence each other in the decision domain; informational relationships about what action sequences are feasible in any given set of circumstances; and functional relationships about how desirable the consequences are [21]. An ID can systematically model all the relevant random variables and decision variables in a compact graphical model.

In the past several years, a few methods have been proposed to compute VOI in IDs. Ezawa [8] introduces some basic concepts about VOI and evidence propagation in IDs. Dittmer and Jensen [7] present a method for calculating myopic VOI in IDs based on the strong junction tree framework [15]. Shachter [25] further improves this method by enhancing the strong junction tree as well as developing methods for reusing the original tree in order to perform multiple VOI calculations. Zhang et al. [28] present an algorithm to speed up the VOI computation by making use of the intermediate computation results, which are obtained when computing the optimal expected value of the original ID without the observations from the information sources. Instead of computing VOI directly, [22] describe a procedure to identify a partial order over variables in terms of their VOIs based on the topological relationships among variables in the ID. However, all these papers only focus on computing myopic VOI, which is based on two assumptions: (1) "No competition:" each information source is evaluated in isolation, as if it were the only source available for the entire decision; (2) "One-step horizon:" the decision maker will act immediately after consulting the source [21]. These assumptions result in a myopic policy: every time, the decision maker evaluates the VOI of each information source one by one, and chooses the one with the largest VOI. Then the observations are collected from the selected information sources, the probabilities are updated, and all the remaining information sources are to be reevaluated again, and a similar procedure repeats.

Obviously, the assumptions are not always reasonable in some decision circumstances. Usually, the decision maker will not act after acquiring only one information source. Also, although a single information source may have low VOI and is not worth acquisition compared to its cost, several information sources used together may have high VOI compared to their combined cost. In this case, by only evaluating myopic VOI, the conclusion will be not to collect such information, which is not optimal since its usage together with other information sources can lead to high value for the decision maker. Therefore, given these limitations in myopic VOI, it is necessary to compute non-myopic VOI.

Non-myopic VOI respects the fact that the decision maker may observe more than one piece of information before acting, thus requires the consideration of any possible ordered sequence of observations given a set of information sources. Unfortunately, the number of the sequences grows exponentially as the number of available information sources increases, and thus it is usually too cumbersome to compute non-myopic VOI for any practical use, and this is why the before mentioned work only focuses on myopic VOI. Given these facts, an approximate computation of non-myopic VOI is necessary to make it feasible in practical applications. To the best of our knowledge, [11] are the only ones who proposed a solution to this problem. In their approach, the central-limit theorem is applied to approximately compute non-myopic VOI in a special type of ID for the diagnosis problem, where only one decision node exists. Certain assumptions are required in their method: (1) all the random nodes and decision nodes in the ID are required to be binary; (2) the information sources are conditionally independent from each other given the hypothesis node, which is the node associated with the decision node and utility node.

Motivated by the method of Heckerman et al., we extend this method to more general cases[1]: (1) all the random nodes can have multiple states and the decision node can have multiple rules (alternatives); (2) the information sources can be dependent given the hypothesis node; and (3) the ID can have a more general structure. But same as Heckerman et al.'s method, we only discuss the VOI computation in terms of IDs that have only one decision node. This decision node shares only one utility node with another chance node. With the proposed algorithm, non-myopic VOI can be efficiently approximated. In order to validate the performance of the proposed algorithm, we not only perform the experiments based on the synthetic data for various types of IDs, but also provide a real-world application with real data.

Because of the efficiency and accuracy of the proposed method, we believe that it can be widely used to choose the optimal set of available information sources for a wide range of applications. No matter what selection strategies people use to choose an optimal set, such as greedy approaches, heuristic searching algorithms, or brute-force methods, the proposed method can be utilized to evaluate any information set efficiently in order to speed up the selection procedure.

The following sections are organized as follows. Section 2 presents a brief introduction to influence diagrams. The detail of the algorithm is described in Section 3. Section 4 discusses the experimental results based on synthetic data. And a real application is demonstrated in Section 5. Finally, Section 6 gives the conclusion and some suggestions for future work.

2. Influence diagrams

An influence diagram (ID) is a graphical representation of a decision-making problem under uncertainty. Its knowledge representation can be viewed through three hierarchical levels, namely, relational, functional, and numerical. At the relational level, an ID represents the relationships between different variables through an acyclic directed graph consisting of various node types and directed arcs. The functional level specifies the interrelationships between various node types and defines the corresponding conditional probability distributions. Finally, the numerical level specifies the actual numbers associated with the probability distributions and utility values [6].

[1] A brief version of this extension can be found in [18].

Specifically, an ID includes three types of nodes: decision, chance (random), and value (utility) nodes. Decision nodes, usually drawn as rectangles, indicate the decisions to be made and their set of possible alternative values. Chance nodes, usually drawn as circles/ellipses, represent uncertain variables that are relevant to the decision problem. They are similar to the nodes in Bayesian networks [14], and are associated with conditional probability tables (CPTs). Value nodes, usually drawn as diamonds, are associated with utility functions to represent the utility of each possible combination of the outcomes of the parent node. The arcs connecting different types of nodes have different meanings. An arc between two chance nodes represents probabilistic dependence, while an arc from a decision node to a chance node represents functional dependence, which means the actions associated with the decision node affect the outcome of the chance node. An arc between two decision nodes implies time precedence, while an arc from a chance node to a decision node is informational, i.e., it shows which variable will be known to the decision maker before a decision is made [21]. An arc pointing to a utility node represents value influence, which indicates that the parents of the utility node are those that directly affect its utility. Fig. 1 illustrates these arcs and gives corresponding interpretations.

Most IDs assume a precedence ordering of the decision nodes. A regular ID assumes that there is a directed path containing all decision nodes; a no-forgetting ID assumes that each decision node and its parents are also parents of the successive decision nodes; and a stepwise decomposable ID assumes that the parents of each decision node divide the ID into two separate fractions. In this paper, we consider IDs that have only one decision node, i.e., ignoring all previous decisions. The goal of ID modeling is to choose an optimal policy that maximizes the overall expected utility. A policy is a sequence of decision rules where each rule corresponds to one decision node. Mathematically, if there is only one decision node in an ID and assuming additive decomposition of the utility functions, the expected utility under a decision rule d given any available evidence e, denoted by $EU(d|e)$, can be defined as follows:

$$EU(d|e) = \sum_{i=1}^{n} \sum_{X_i} p(X_i|e,d)u_i(X_i,d), \tag{1}$$

where u_i is the utility function over the domain $X_i \cup \{D\}$. For example, X_i could be the parents of the utility node that u_i is associated with. To evaluate an ID is to find an optimal policy as well as to compute its optimal expected utility [24,26]. More detail about IDs can be found in [17,14].

Generally, the advantages of an ID can be summarized by its compact and intuitive formulation, its easy numerical assessment, and its effective graphical representation of dependence between variables for modeling decision making under uncertainty. These benefits make ID a widely used tool to model and solve complex decision problems in recent years.

3. Approximate VOI computation

3.1. Value of information

The VOI of a set of information sources is defined as the difference between the maximum expected utilities with and without the information sources [17]. VOI can be used to rate the usefulness of various information sources and to decide whether pieces of evidence are worth acquisition before actually using the information sources [21].

We discuss the VOI computation in terms of IDs that have only one decision node. This decision node shares only one utility node with another chance node, as shown in Fig. 2. And the decision node and the chance node are assumed to be independent. In the ID, the chance node Θ, named as hypothesis node, represents a mutually exclusive and exhaustive set of possible hypotheses $\theta_1, \theta_2, \ldots, \theta_h$; the decision node D represents a set of possible alternatives $d_1, d_2, \ldots, d_q$; the utility node U represents the utility of the decision maker, which depends on the outcome of Θ and D; and the chance nodes $O_1, \ldots, O_n$ represent possible observations from all kinds of information sources about the true state of Θ. And each O_i may have multiple states. Let $O = \{O_1, \ldots, O_n\}$, the VOI of O, VOI(O), w.r.t. the decision node D, can be defined as follows:

$$\mathrm{VOI}(O) = EU(O) - EU(\overline{O}), \tag{2}$$

$$EU(O) = \sum_{o \in O} p(o) \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i|o)u(\theta_i, d_j), \tag{3}$$

$$EU(\overline{O}) = \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i)u(\theta_i, d_j), \tag{4}$$

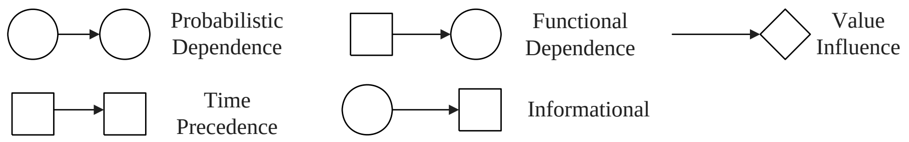


Fig. 1. Interpretations of arcs in an ID, where circles represent chance (random) nodes, rectangles for decision nodes, and diamonds for value (utility) nodes.

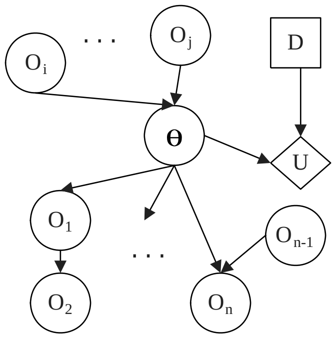


Fig. 2. An ID example for non-myopic VOI computation. Θ is the hypothesis node, D is the decision node, and U is the utility node. O_i represents possible observations from an information source. There could be hidden nodes between Θ and O_i.

where $u()$ denotes the utility function associated with the utility node U, $EU(O)$ denotes the expected utility to the decision maker if O were observed, while $EU(\overline{O})$ denotes the expected utility to the decision maker without observing O. Here the cost of collecting information from the information sources is not included; thus, the VOI can also be called perfect VOI [11]. The net VOI is the difference between the perfect VOI and the cost of collecting information [12]. Since after calculating the perfect VOI, the computation of the net VOI is just a subtraction of cost, we focus on the perfect VOI in the subsequent sections.

As shown in Eq. (2), to compute VOI(O), it is necessary to compute $EU(O)$ and $EU(\overline{O})$ respectively. Obviously, $EU(\overline{O})$ is easier to compute, whereas directly computing $EU(O)$ could be cumbersome. If the decision maker has the option to observe a subset of observations $\{O_1, \ldots, O_n\}$ and each O_i has m possible values, then there are m^n possible instantiations of the observations in this set. Thus, to compute $EU(O)$, there are m^n inferences to be performed. In other words, the time complexity of computing VOI is exponential. It becomes infeasible to compute VOI(O) when n is not small.

The key to computing VOI(O) efficiently is to compute $EU(O)$, which can be rewritten as follows:

$$EU(O) = \sum_{o \in O} p(o) \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i|o)u(\theta_i, d_j) = \sum_{o \in O} \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(o)p(\theta_i|o)u(\theta_i, d_j) = \sum_{o \in O} \max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i)p(o|\theta_i)u(\theta_i, d_j). \tag{5}$$

It is noticed that each instantiation of O corresponds to a specific optimal action for the decision node D. We define the decision function $\delta : O \to D$, which maps an instantiation of O into a decision in D. For example, $\delta(o) = d_k$ indicates when the observation is o, the corresponding optimal decision is d_k, $d_k = \arg\max_{d_j \in D} \sum_{\theta_i \in \Theta} p(\theta_i|o)u(\theta_i, d_j)$. Therefore we can divide all the instantiations of O into several subsets, where the optimal action is the same for those instantiations in the same subset. Specifically, if D has q decision rules, $\{d_1, \ldots, d_q\}$, all the instantiations of O can be divided into q subsets, $o_{d_1}, o_{d_2}, \ldots, o_{d_q}$, where $o_{d_k} = \{o \in O | \delta(o) = d_k\}$. Fig. 3 illustrates the relationships between each instantiation and the q subsets. Thus, from Eq. (5), $EU(O)$ can be further derived as follows:

$$EU(O) = \sum_{\theta_i \in \Theta} p(\theta_i) \sum_{k=1}^{q} \sum_{o \in o_{d_k}} p(o|\theta_i)u(\theta_i, d_k). \tag{6}$$

In the next several sections, we show how to compute $EU(O)$ efficiently.

3.2. Decision boundaries

In Eq. (6), the difficult part is to compute $\sum_{o \in o_{d_k}} p(o|\theta_i)$ because the size of the set o_{d_k} could be very large based on the previous analysis. In order to compute it efficiently, it is necessary to know how to divide all the instantiations of O into the q subsets. We first focus on the case that Θ has only two states, θ_1, θ_2, and then extend it to the general case in Section 3.4.

Based on the definition, the expected utility of taking the action d_k is $EU(d_k) = p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k}$, where $u_{1k} = u(\theta_1, d_k)$, and $u_{2k} = u(\theta_2, d_k)$. We can sort the index of all the decision rules based on the utility functions, such that $u_{1k} > u_{1j}$ and $u_{2k} < u_{2j}$ for $k < j$. Fig. 4 gives an example of the utility function $u(\Theta, D)$. As shown in the figure, as k increases, u_{1k} decreases and u_{2k} increases. If there is an action d_i that cannot be sorted according to this criterion, it is either

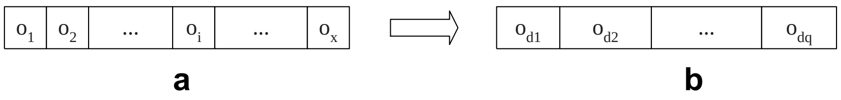


Fig. 3. (a) Each o_i corresponds to an instantiation; (b) all the instantiations can be divided into q subsets, where each instantiation in the set o_{d_i} corresponds to the optimal decision d_i.

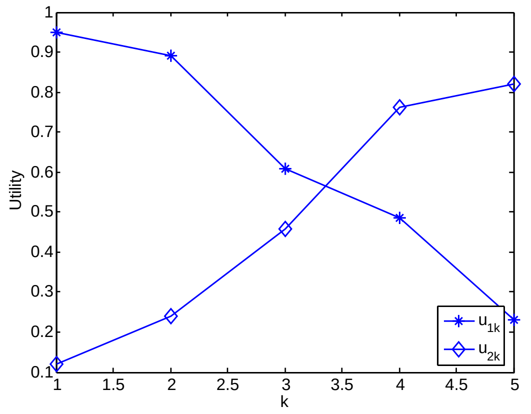


Fig. 4. An example of the utility function $U(\Theta, D)$.

dominated by another action, or it dominates another action. (If $u(d_i, \Theta)$ is always larger than $u(d_j, \Theta)$, no matter what state of Θ is, we say d_i dominates d_j). Then the dominated action can be removed from the set of possible actions, without changing the optimal policy.

Proposition 1. *Let* $r_{jk} = \frac{u_{2j}-u_{2k}}{u_{1k}-u_{1j}+u_{2j}-u_{2k}}$, $p^*_{kl} = \max_{k<j\leqslant q} r_{jk}$, *and* $p^*_{ku} = \min_{1\leqslant j<k} r_{jk}$, *then* d_k *is the optimal action if and only if* $p^*_{kl} \leqslant p(\theta_1) \leqslant p^*_{ku}$. *In addition,* $p^*_{ql} = 0$ *and* $p^*_{1u} = 1$. *(Here k is the index of an action.)*

Proof. see Appendix. □

Proposition 1 presents that if the probability of Θ being θ_1 is between p^*_{kl} and p^*_{ku}, d_k is the optimal decision. From this, we can further derive Proposition 2.

Proposition 2

$$\sum_{o\in o_{d_k}} p(o) = p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}). \tag{7}$$

Proof. see Appendix. □

The Proof of Proposition 2 establishes Eq. (7) by showing that both sides of this equation express the probability that d_k is the optimal decision for θ_1. Based on Proposition 2, we can get the following corollary.

Corollary 1

$$\sum_{o\in o_{d_k}} p(o|\theta_1) = p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1), \tag{8}$$

$$\sum_{o\in o_{d_k}} p(o|\theta_2) = p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_2). \tag{9}$$

The equations in Corollary 1 indicate the probability that the decision maker will take the optimal decision d_k after observing new evidence, given the situation that the state of Θ is θ_i before collecting the evidence.

Based on Corollary 1, the problem of computing $\sum_{o\in o_{d_k}} p(o|\theta_i), i = 1, 2$, (from Eq. (6)) transfers to the problem of computing $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_i)$, which is the topic of the next section. We will focus on $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1)$ only because the procedure of computing $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_2)$ is similar.

3.3. Approximation with central-limit theorem

3.3.1. A partitioning procedure

To compute $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1)$, one way is to treat $p(\theta_1|o)$ as a random variable. If the probability density function of this variable is known, it will be easy to compute $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1)$. However, it is hard to get such a probability density function directly. But we notice that $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1) = p\left(\frac{p^*_{kl}}{1-p^*_{kl}} \leqslant \frac{p(\theta_1|o)}{p(\theta_2|o)} \leqslant \frac{p^*_{ku}}{1-p^*_{ku}}|\theta_1\right)$. Based on the transformation property between a random variable and its function [2], it is straightforward that $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku}|\theta_1)$ $= p\left(\frac{p^*_{kl}}{1-p^*_{kl}} \leqslant \frac{p(\theta_1|o)}{p(\theta_2|o)} \leqslant \frac{p^*_{ku}}{1-p^*_{ku}}|\theta_1\right)$.

Let us take a closer look at $\frac{p(\theta_1|o)}{p(\theta_2|o)}$ because it is critical in the approximate algorithm.

If all the O_i nodes are conditionally independent from each other given Θ, based on the chain rule:

$$\frac{p(\theta_1|O)}{p(\theta_2|O)} = \frac{p(O_1|\theta_1)}{p(O_1|\theta_2)} \cdots \frac{p(O_n|\theta_1)}{p(O_n|\theta_2)} \frac{p(\theta_1)}{p(\theta_2)}. \tag{10}$$

Usually some O_is may not be conditionally independent given Θ. We will show that $\frac{p(\theta_1|o)}{p(\theta_2|o)}$ is approximately distributed as a log-normal random variable. However, in order to prove it, it is necessary to obtain a format similar to Eq. (10) even when O_is are not conditionally independent. We thus propose a partitioning procedure to partition O into several groups based on the principle of *d*-separation [21], where the nodes in one group are conditionally independent from the nodes in other groups. This procedure consists of three steps.

(1) Decide whether two nodes, O_i, O_j, are conditionally independent given Θ by exploring the ID structure based on four rules: (i) if there is a **directed** path between O_i and O_j without passing Θ, O_i and O_j are dependent; (ii) if both O_i and O_j are the ancestors of Θ, O_i and O_j are dependent given Θ; (iii) after removing the links to and from Θ from the original ID, if O_i and O_j have common ancestors, or O_i is O_j's ancestor, or vice versa, then O_i and O_j are dependent; and (iv) in all the other cases, O_i and O_j are conditionally independent given Θ.
(2) Build an undirected graph to model the relationships between the nodes. In such a graph, each vertex represents an O_i node, and each edge between two vertices indicates that the two corresponding nodes are dependent according to the rules in Step 1.
(3) Partition the graph into disjoint connected subgraphs. A depth first search (DFS) algorithm [4] is used to partition the graph into several *connected components* (disjoint connected subgraphs) so that each component is disconnected from other components. The nodes in each connected component are conditionally independent from the nodes in any other connected components. Therefore, each connected component corresponds to one group.

For example, for the ID in Fig. 5a, with the partitioning procedure, the O_i nodes can be divided into five groups, $\{O_1, O_2\}$, $\{O_3, O_4, O_5\}$, $\{O_6\}$, $\{O_7\}$, and $\{O_8, O_9\}$. Fig. 5b shows the graph built by the partitioning procedure.

3.3.2. Central-limit theorem

Generally, with the partition procedure presented in the previous subsection, O can be automatically divided into several sets, named $O^{s_1}, O^{s_2}, \ldots, O^{s_g}$, where g is the overall number of the groups. Thus, Eq. (10) can be modified as follows:

$$\frac{p(\theta_1|O)}{p(\theta_2|O)} = \frac{p(O^{s_1}|\theta_1)}{p(O^{s_1}|\theta_2)} \cdots \frac{p(O^{s_g}|\theta_1)}{p(O^{s_g}|\theta_2)} \frac{p(\theta_1)}{p(\theta_2)} \Rightarrow \ln\frac{p(\theta_1|O)}{p(\theta_2|O)} = \sum_{i=1}^{g} \ln\frac{p(O^{s_i}|\theta_1)}{p(O^{s_i}|\theta_2)} + \ln\frac{p(\theta_1)}{p(\theta_2)} \Rightarrow \ln\phi = \sum_{i=1}^{g} w_i + c,$$
$$\text{where } \phi = \frac{p(\theta_1|O)}{p(\theta_2|O)}, \quad w_i = ln\frac{p(O^{s_i}|\theta_1)}{p(O^{s_i}|\theta_2)}, \quad c = \ln\frac{p(\theta_1)}{p(\theta_2)}. \tag{11}$$

In the above equation, c can be regarded as a constant reflecting the state of Θ before any new observation is obtained and any new decision is taken. Here, we assume $p(\theta_2|O)$, $p(O^{s_i}|\theta_2)$, and $p(\theta_2)$ are not equal to 0.

Let $W = \sum_{i=1}^{g} w_i$ be the sum of w_i. Following [11], we use the cental-limit theorem to approximate W. The central-limit theorem [9] states that the sum of independent variables approaches a Gaussian distribution when the number of variables becomes large. Also, the expectation and variance of the sum is the sum of the expectation and variance of each individual random variable. Thus, regarding each w_i as an independent variable, W then follows a Gaussian distribution. Then, based on Eq. (11), ϕ will be a log-normal distribution. For a random variable X, if $\ln(X)$ has a Gaussian distribution, we say X has a log-normal distribution. The probability density function is: $p(x) = \frac{1}{S\sqrt{2\pi}x} e^{-(\ln x - M)^2/(2S^2)}$, denoted as $X \sim \mathrm{LogN}(M, S^2)$ [5], where M and S are the mean and standard deviation of the variable's logarithm [1]. In order to assess the parameters (mean and var-

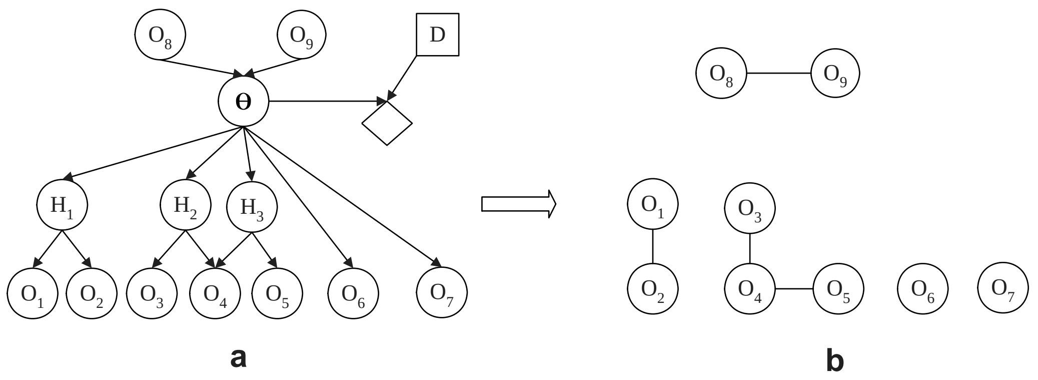


Fig. 5. (a) An ID example; (b) the graph built by the partitioning procedure.

iance) of the log-normal distribution, we need to compute the mean and the variance of each w_i. The computational process is shown as follows.

Assume O^{s_i} has r_i instantiations, $\{o_1^{s_i}, \dots, o_{r_i}^{s_i}\}$, where r_i is the product of the number of the states for each node in the group O^{s_i}, e.g., if $O^{s_i} = \{O_1, O_2\}$, and both O_1 and O_2 have three states, then $r_i = 3 * 3 = 9$. Table 1 gives the value and the probability distribution for each w_i:

Based on the table, the expected value μ, and the variance σ^2 for each w_i can be computed as follows:

$$\mu(w_i|\theta_1) = \sum_{j=1}^{r_i} p(o_j^{s_i}|\theta_1) ln \frac{p(o_j^{s_i}|\theta_1)}{p(o_j^{s_i}|\theta_2)}, \tag{12}$$

$$\sigma^2(w_i|\theta_1) = \sum_{j=1}^{r_i} p(o_j^{s_i}|\theta_1) ln^2 \frac{p(o_j^{s_i}|\theta_1)}{p(o_j^{s_i}|\theta_2)} - \mu^2(w_i|\theta_1). \tag{13}$$

By the central-limit theorem, the expected value and the variance of W can be obtained by the following equations:

$$\mu(W|\theta_1) = \sum_{i=1}^{g} \mu(w_i|\theta_1), \tag{14}$$

$$\sigma^2(W|\theta_1) = \sum_{i=1}^{g} \sigma^2(w_i|\theta_1). \tag{15}$$

Therefore, based on Eq. (11), for $W \sim N(\mu(W|\theta_1), \sigma^2(W|\theta_1))$, we have $\phi \sim \text{LogN}(\mu(W|\theta_1) + c, \sigma^2(W|\theta_1))$, where LogN denotes the log-normal distribution. After getting the probability distribution function and the function parameters for ϕ in Eq. (11), we are ready to assess the non-myopic VOI.

Before we go to the next section, we first analyze the computational steps involved in computing the parameters for the log-normal distribution, which is the most time-consuming part in the algorithm. Based on Eqs. (12) and (14), the overall number of the computational steps is $4\sum_{i=1}^{g} r_i + 2g$. We will show that this number is much smaller than the overall number of the computational steps in the exact computational method during the algorithm analysis in Section 3.5.

3.3.3. *Approximate non-myopic value-of-information*

Based on Proposition 1 in Section 3.2, we know that d_k is the optimal action with the probability $p(p_{kl}^* \leqslant p(\theta_1|o) \leqslant p_{ku}^*)$, which is equivalent to $p\left(\frac{p_{kl}^*}{1-p_{kl}^*} \leqslant \phi \leqslant \frac{p_{ku}^*}{1-p_{ku}^*}\right)$ as shown in Section 3.3.1. Let $\phi_{kl}^* = \frac{p_{kl}^*}{1-p_{kl}^*}$, and $\phi_{ku}^* = \frac{p_{ku}^*}{1-p_{ku}^*}$, thus, d_k is the optimal decision if and only if $\phi_{kl}^* \leqslant \phi \leqslant \phi_{ku}^*$. Then, based on Corollary 1 in Section 3.2, the following equation stands:

$$\sum_{o \in o_{d_k}} p(o|\theta_1) = p(\phi_{kl}^* \leqslant \phi \leqslant \phi_{ku}^*|\theta_1). \tag{16}$$

Furthermore, from Section 3.3.2, we know that $\phi \sim \text{LogN}(\mu(W|\theta_1) + c, \sigma^2(W|\theta_1))$, thus,

$$p(\phi_{kl}^* \leqslant \phi \leqslant \phi_{ku}^*|\theta_1) = \frac{1}{\sigma(W|\theta_1)\sqrt{2\pi}x} \int_{\phi_{kl}^*}^{\phi_{ku}^*} e^{\frac{-(\ln x - \mu(W|\theta_1) - c)^2}{2\sigma^2(W|\theta_1)}} dx, \tag{17}$$

$p(\phi_{kl}^* \leqslant \phi \leqslant \phi_{ku}^*|\theta_2)$ can be computed in the same way by replacing θ_1 with θ_2 in the previous equations.

Therefore, VOI can be approximated by combining Eqs. (2), (6), (16), and (17). Fig. 6 shows the key equations of the algorithm when Θ has only two states. In summary, to approximate VOI(O) efficiently, the key is to compute $EU(O)$, which leads to an approximation of $\sum_{o \in o_{d_k}} p(o|\theta_1)$ with the log-normal distribution by exploiting the central-limit theorem and the decision boundaries.

3.4. *Generalization*

In the previous algorithm, the node Θ only allows two states, although the other random nodes and the decision node can be multiple states. However, in real-world applications, Θ may have more than two states. In this section, we extend the algorithm to the case that Θ can have several states too. Assume Θ has h states, $\theta_1, \dots, \theta_h$, and still, d has q rules, $d_1, \dots, d_q$, similarly to Eq. (11), we have the following equations:

Table 1
The probability distribution of w_i

w_i	$p(w_i\|\theta_1)$	$p(w_i\|\theta_2)$
$\ln \frac{p(o_1^{s_i}\|\theta_1)}{p(o_1^{s_i}\|\theta_2)}$	$p(o_1^{s_i}\|\theta_1)$	$p(o_1^{s_i}\|\theta_2)$
...	...	...
$\ln \frac{p(o_{r_i}^{s_i}\|\theta_1)}{p(o_{r_i}^{s_i}\|\theta_2)}$	$p(o_{r_i}^{s_i}\|\theta_1)$	$p(o_{r_i}^{s_i}\|\theta_2)$

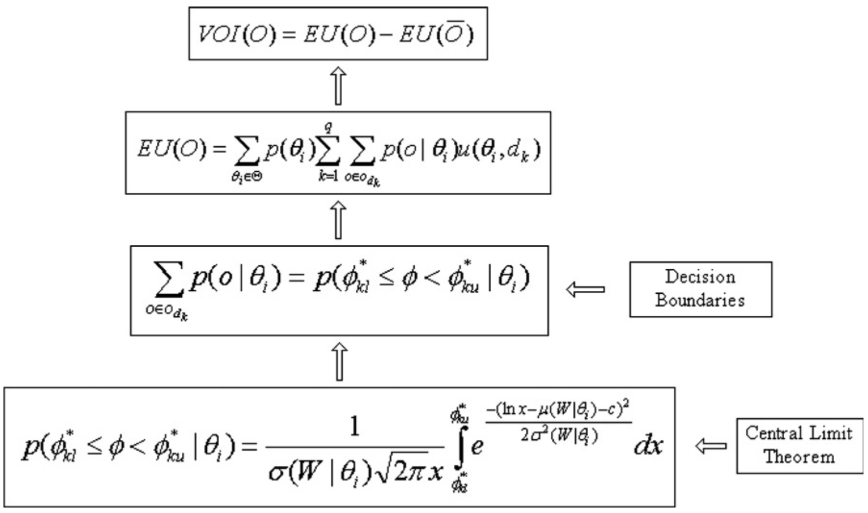

$VOI(O) = EU(O) - EU(\bar{O})$
$EU(O) = \sum_{\theta_i \in \Theta} p(\theta_i) \sum_{k=1}^{q} \sum_{o \in o_{d_k}} p(o \mid \theta_i) u(\theta_i, d_k)$
$\sum_{o \in o_{d_k}} p(o \mid \theta_i) = p(\phi_{kl}^* \le \phi < \phi_{ku}^* \mid \theta_i)$
Decision Boundaries
$p(\phi_{kl}^* \le \phi < \phi_{ku}^* \mid \theta_i) = \frac{1}{\sigma(W \mid \theta_i)\sqrt{2\pi}x} \int_{\phi_{kl}^*}^{\phi_{ku}^*} e^{\frac{-(\ln x - \mu(W|\theta_i) - c)^2}{2\sigma^2(W|\theta_i)}} dx$
Central Limit Theorem

Fig. 6. The key equations to approximate VOI when Θ has only two states, D has multiple rules, and the other nodes have multiple states.

$$\frac{p(\theta_i|O)}{p(\theta_h|O)} = \frac{p(O^{s_1}|\theta_i)}{p(O^{s_1}|\theta_h)} \cdots \frac{p(O^{s_g}|\theta_i)}{p(O^{s_g}|\theta_h)} \frac{p(\theta_i)}{p(\theta_h)}, \quad i \neq h \Rightarrow \ln \frac{p(\theta_i|O)}{p(\theta_h|O)} = \sum_{k=1}^{g} \ln \frac{p(O^{s_k}|\theta_i)}{p(O^{s_k}|\theta_h)} + \ln \frac{p(\theta_i)}{p(\theta_h)} \Rightarrow \ln \phi_i$$
$$= \sum_{k=1}^{g} w_k^i + c_i, \quad \text{where } \phi_i = \frac{p(\theta_i|O)}{p(\theta_h|O)}, \quad w_k^i = \ln \frac{p(O^{s_k}|\theta_i)}{p(O^{s_k}|\theta_h)}, \quad c_i = \ln \frac{p(\theta_i)}{p(\theta_h)}. \tag{18}$$

Let $W_i = \sum_{k=1}^{g} w_k^i$, $i \neq h$, W_i still has a Gaussian distribution. Here, we assume $p(\theta_h|O)$, $p(O^{s_k}|\theta_h)$, and $p(\theta_h)$ are not equal to 0. The similar method in Section 3.3 can be used to compute the variance and the mean. Specifically, for the new defined w_k^i in the above equation, Table 1 can be modified as follows (see Table 2).

Thus, we get the following equations:

$$\mu(w_k^i|\theta_j) = \sum_{l=1}^{r_k} p(o_l^{s_k}|\theta_j) \ln \frac{p(o_l^{s_k}|\theta_i)}{p(o_l^{s_k}|\theta_h)}, \quad 1 \leqslant i < h, \ 1 \leqslant j \leqslant h, \ 1 \leqslant k \leqslant g, \tag{19}$$

$$\sigma^2(w_k^i|\theta_j) = \sum_{l=1}^{r_k} p(o_l^{s_k}|\theta_j) \ln^2 \frac{p(o_l^{s_k}|\theta_i)}{p(o_l^{s_k}|\theta_h)} - \mu^2(w_k^i|\theta_j). \tag{20}$$

Similar to Eq. (14), the expected value and the variance of W_i can be obtained as we see here:

$$\mu(W_i|\theta_j) = \sum_{k=1}^{g} \mu(w_k^i|\theta_j), \quad 1 \leqslant i < h, \ 1 \leqslant j \leqslant h, \tag{21}$$

$$\sigma^2(W_i|\theta_j) = \sum_{k=1}^{g} \sigma^2(w_k^i|\theta_j). \tag{22}$$

Accordingly, ϕ_i follows the log-normal distribution with $S_{ij} = \sigma(W_i|\theta_j)$ and $M_{ij} = \mu(W_i|\theta_j) + c_i$. We denote the probability density function of ϕ_i given θ_j as $f_{\theta_j}(\phi_i)$. Eqs. (19) and (21) show that the overall number of the computational steps to assess the parameters for the log-normal distributions is $4h\sum_{k=1}^{g} r_k + 2h(h-1)g$ when $h > 2$.

Even though $f_{\theta_j}(\phi_i)$ can be easily obtained, it is still necessary to get the decision boundaries for each optimal decision in order to efficiently compute $\sum_{o \in o_{d_k}} p(o|\theta_j)$. Therefore, a set of linear inequality functions need to be solved when Θ has more than two states. For example, if d_k is the optimal action, $EU(d_k)$ must be larger than the expected utility of taking any other action. Based on this, a set of linear inequality functions can be obtained:

Table 2
The probability distribution of w_k^i

w_k^i	$p(w_k^i\|\theta_1)$	...	$p(w_k^i\|\theta_h)$
$\ln \frac{p(o_1^{s_k}\|\theta_i)}{p(o_1^{s_k}\|\theta_h)}$	$p(o_1^{s_k}\|\theta_1)$	...	$p(o_1^{s_k}\|\theta_h)$
...	...	...	...
$\ln \frac{p(o_{r_k}^{s_k}\|\theta_i)}{p(o_{r_k}^{s_k}\|\theta_h)}$	$p(o_{r_k}^{s_k}\|\theta_1)$	...	$p(o_{r_k}^{s_k}\|\theta_h)$

$$\begin{aligned} p(\theta_1)u_{1k} + p(\theta_2)u_{2k} + \cdots + p(\theta_h)u_{hk} &\geqslant p(\theta_1)u_{1j} + \cdots + p(\theta_h)u_{hj} \\ &\Rightarrow \frac{u_{1k} - u_{1j} + u_{hj} - u_{hk}}{u_{hj} - u_{hk}} \cdot p(\theta_1) + \cdots + \frac{u_{(h-1)k} - u_{(h-1)j} + u_{hj} - u_{hk}}{u_{hj} - u_{hk}} \cdot p(\theta_{h-1}) \geqslant 1 \\ &\Rightarrow \frac{u_{1k} - u_{1j}}{u_{hj} - u_{hk}} \cdot \frac{p(\theta_1)}{p(\theta_h)} + \cdots + \frac{u_{(h-1)k} - u_{(h-1)j}}{u_{hj} - u_{hk}} \cdot \frac{p(\theta_{h-1})}{p(\theta_h)} \geqslant 1. \end{aligned} \tag{23}$$

We assume $u_{hj} - u_{hk} > 0$; otherwise, "$\geqslant$" is changed to "$\leqslant$" in the last inequality.

Let A_k be the solution region of the above linear inequalities, then

$$\sum_{o \in o_{d_k}} p(o|\theta_j) = \int_{A_k} f_{\theta_j}(\phi_1) \cdots f_{\theta_j}(\phi_{h-1}) \, dA_k, \quad 1 \leqslant j \leqslant h, \ 1 \leqslant k \leqslant q. \tag{24}$$

The right side of Eq. (24) is an integral over the solution region A_k decided by the linear inequalities. We first demonstrate how to solve the integral when Θ has three states, and then introduce the method for the case that Θ has more than three states.

When Θ has three states, Eq. (23) can be simplified as follows:

$$\begin{aligned} &p(\theta_1)u_{1k} + p(\theta_2)u_{2k} + p(\theta_3)u_{3k} \geqslant p(\theta_1)u_{1j} + p(\theta_2)u_{2j} + p(\theta_3)u_{3j} \Rightarrow \alpha_{1kj} \cdot \frac{p(\theta_1)}{p(\theta_3)} + \alpha_{2kj} \cdot \frac{p(\theta_2)}{p(\theta_3)} \geqslant 1, \\ &\text{where } \alpha_{1kj} = \frac{u_{1k} - u_{1j}}{u_{3j} - u_{3k}} \text{ and } \alpha_{2kj} = \frac{u_{2k} - u_{2j}}{u_{3j} - u_{3k}}. \end{aligned} \tag{25}$$

In the above, it is assumed that $u_{3j} > u_{3k}$; if $u_{3j} < u_{3k}$, then "$\geqslant$" is changed to "$\leqslant$" in the last inequality.

And Eq. (24) can be simplified as follows:

$$\sum_{o \in o_{d_k}} p(o|\theta_j) = \int_{A_k} f_{\theta_j}(\phi_1) f_{\theta_j}(\phi_2) \, dA_k, \quad 1 \leqslant k \leqslant q, \ 1 \leqslant j \leqslant 3, \tag{26}$$

A_k is decided by $(q-1)$ linear inequalities and each inequality has two variables ϕ_1 and ϕ_2 as defined in Eq. (25). We use the following steps to solve this integral when A_k is a finite region.

1. Identify all the lines that define the inequalities and find all the intersection points between any two lines as well as the intersection points between any line and the x (or y) axis.
2. Choose the intersection points that satisfy all the linear inequalities, and use them as vertices to form a polygon.
3. Divide the polygon into several simple regions:Specifically, for each vertex, we generate a line crossing this vertex and parallel to the y-axis. The lines then divide the polygon into several simple regions.
4. Evaluate the integral in each simple region and sum the values together.

An example of the solution region is shown in Fig. 7. In this example, if $\alpha_{1kj} > \alpha_{1kj} (i \neq j)$, then $\alpha_{2kj} > \alpha_{2kj}$ too. Therefore, the solution region can be decided by the intersection points of the lines that are defined by the linear inequalities and the axes. For example, in Fig. 7, A_k is decided by a–d, which are selected from the intersection points $\{(1/\alpha_{1kj}, 0), (0, 1/\alpha_{2kj}), j = 1, \ldots, q, j \neq k\}$. Based on [3], the time complexity of solving m linear inequalities with n variables (each inequality only has two variables) is $O(mn \log m + mn^2 \log^2 n)$. In this case, n is 2 and m is $q-1$.

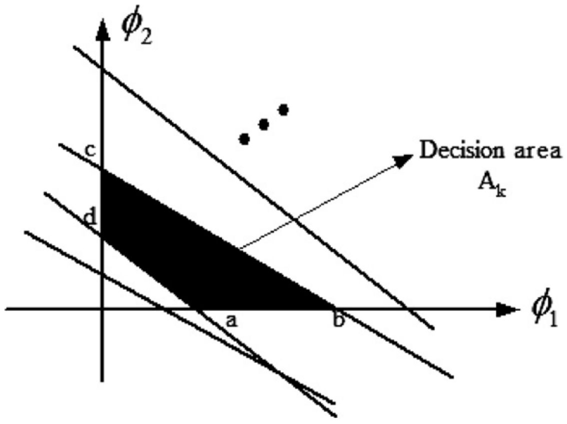


Fig. 7. A solution region of a group of linear inequalities.

When Θ has more than three states, the integral needs to be performed in a high-dimension space (dimension is larger than 2). Therefore, we solve it with Quasi-Monte Carlo integration [10,16], which is a popular method to handle multiple integral. Quasi-Monte Carlo integration picks points based on sequences of quasirandom numbers over some simple domain A'_k which is a superset of A_k, checks whether each point is within A_k, and estimates the area (n-dimensional content) of A_k as the area of A'_k multiplied by the fraction of points falling within A_k. Such a method is implemented by Mathematica [20], which can automatically handle a multiple integral with a region implicitly defined by multiple inequality functions.

Fig. 8 shows the key equations of the algorithm when Θ has multiple states. The main equations are similar to those in Fig. 6. However, since Θ has multiple states, it becomes more complex to obtain the parameters of the log-normal distribution and perform the integration.

3.5. Algorithm analysis

Now, we analyze the computational complexity of the proposed approximation algorithm compared to the exact computational method. For simplicity, assume that the number of the state of each O_i node is m, and there are n nodes in the set O. Assume we only count the time used for computing expected utilities. Then the computational complexity of the exact VOI computational method is approximately hm^n, where h is the number of the state of the Θ node. With the approximation algorithm, the computational complexity is reduced to hm^k, where h is the number of the state of the Θ node, and k is the number of O_i nodes in the maximum group among $\{O^{s_1}, \ldots, O^{s_g}\}$. In the best case, if all the O_i nodes are conditionally independent given Θ, the time complexity is about linear with respect to m. In the worst case, if all the O_i nodes are dependent, the time complexity is approximately m^n. However, usually, in most real-world applications, k is less than n, thus, the approximate algorithm is expected to be more efficient than the exact computational method, as will be shown in the experiments. For example, for the ID in Fig. 5, $n = 9$, $m = 4$, $h = 3$, and $q = 3$. Then, for the exact computation, the number of computations is around $3 * 4^9 = 786432$, while using the approximate algorithm, the number of computations is only around $3 * 4^3 = 192$.

However, in addition to the cost of computing expected utilities, the approximation algorithm also includes some extra costs: sorting the utility functions (Section 3.2), partitioning the O set (Section 3.3.1), and deciding the decision boundaries (Section 3.2) when Θ has two states, or performing the integral when Θ has more than two states (Section 3.4). These costs are not included in the above analysis. In general, the extra time in these steps is much less than the time used for computing expected utilities. For example, the time complexity of sorting is $O(q\log(q))$, the time complexity of the partition procedure is $O(|V| + |E|)$ (V is the set of vertex, and E is the set of edges in an ID), and the time complexity in deciding the decision boundaries when θ has two states is $O(q^2)$. When θ has more than two states, deciding the decision boundaries needs additional time. Empirically, it does not affect the overall speed, as will be shown in the experiments. In addition, most steps in computing expected utilities involve performing inferences in an ID, which is usually NP-hard and thus consumes much more time than a step in the procedures of sorting, partitioning, and integrating.

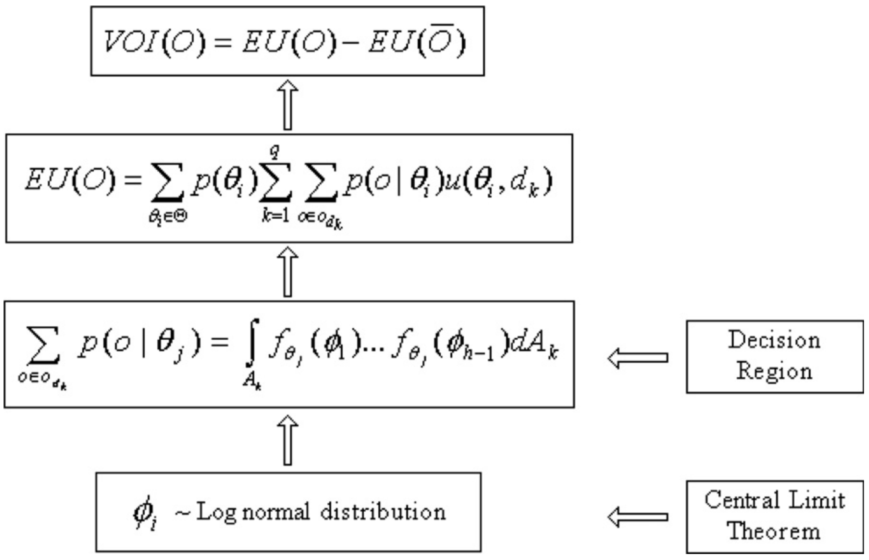

$VOI(O) = EU(O) - EU(\bar{O})$
$EU(O) = \sum_{\theta_i \in \Theta} p(\theta_i) \sum_{k=1}^{q} \sum_{o \in o_{d_k}} p(o \mid \theta_i) u(\theta_i, d_k)$
$\sum_{o \in o_{d_k}} p(o \mid \theta_j) = \int_{A_k} f_{\theta_j}(\phi_1) \ldots f_{\theta_j}(\phi_{h-1}) dA_k$
Decision Region
ϕ_i ~ Log normal distribution
Central Limit Theorem

Fig. 8. The key equations to compute VOI when Θ has multiple states.

Table 3
ID structures

k	5	4	3	2	1
Number of IDs	2	3	3	1	1

k is the size of the biggest group after partitioning.

Table 4
Testing cases

ID_indep: 2-state	50 test cases, where O_i nodes are conditionally independent given Θ whose state is binary
ID_indep: 3-state	50 test cases, where O_i nodes are conditionally independent given Θ who has three states
ID_indep: 4-state	50 test cases, where O_i nodes are conditionally independent given Θ who has four states
ID_dep: 2-state	450 test cases, where O_i nodes are conditionally dependent given Θ whose state is binary
ID_dep: 3-state	450 test cases, where O_i nodes are conditionally dependent given Θ who has three states
ID_dep: 4-state	450 test cases, where O_i nodes are conditionally dependent given Θ who has four states

4. Experiments

The experiments are designed to demonstrate the performance of the proposed algorithm compared to the exact VOI computation. We limit the ID test model with at most 5 layers[2] and up to 11 information sources due to the exponential computational time behind the exact computation. Ten different ID models are constructed, where in one of the IDs the O nodes are conditionally independent given the Θ node. Table 3 describes the structures of these IDs. The IDs are parameterized with 150 sets of different conditional probability tables and utility functions, a process which yields 1500 test cases. In each the one-third of them, Θ node has 2, 3, and 4 states, respectively. Without loss of generality, all the other random nodes and the decision node have four states.

For each test case, the VOIs for different O subsets with the size from 3 to 11 are computed. The results from the approximation algorithm are compared to the exact computation implemented with the brute-forth method. Let VOIt be the ground-truth, and VOI be the value computed with the proposed algorithm. Assuming VOIt $\neq 0$, the error rate is defined as follows:

$$Err = \frac{|\text{VOIt} - \text{VOI}|}{\text{VOIt}}.$$

The 1500 test cases described previously are divided into six groups, named as *ID_indep: 2-state*, *ID_indep:3-state*, *ID_indep:4-state*, *ID_dep:2-state*, *ID_dep: 3-state*, and *ID_dep:4-state*. Table 4 describes the six groups.

Fig. 9 illustrates the results from the six groups of 1500 test cases. Chart (a) shows the average errors for each group, while Chart (b) shows the VOIs for one specific case, which is randomly chosen from the test cases from *ID_dep: 3-state*. As the set size of the O_i nodes increases, the error rate decreases. When the state number of Θ is the same, the error rates of the dependent cases are larger than the error rates of the conditional independent cases. This can be explained by the reason that the IDs in the dependent cases have fewer independent O subsets than the ID in the independent groups. Since the central-limit theorem is the basis of our algorithm, it works better when the number of w_i increases, which corresponds to the number of independent O subsets. Even when the size of O set is as small as 6, the average error is less than or around 0.1 for all the cases. We could run several larger IDs with much more O_i nodes, and the error curve would be progressively decreasing. Here, we intend to show the trend and the capability of this algorithm.

Charts (c) and (d) show the average computational time with the exact computation and the approximation computation. When the set size of the O_i nodes is small, the computational time is similar. However, as the size becomes larger, the computational time of the exact computation increases exponentially, while the computational time of the approximation algorithm increases much slower. Thus, the larger the O set size is, the more time the approximation algorithm can save. Likewise, as the number of the state of each O_i node further increases, the computational saving would be more significant. As the number of states of Θ increase, the computational time also slightly increases.

5. An illustrative application

We use a real-world application in human computer interaction to demonstrate the advantages of the proposed algorithm. Fig. 10 shows an ID for user stress recognition and user assistance. The diagram consists of two portions. The upper portion, from the top to the "stress" node, depicts the elements that can alter human stress. These elements include the workload, the environmental context, specific character of the user such as his/her trait, and importance of the goal that

[2] The length of the longest path starting from (or ending at) the hypothesis node is 5.

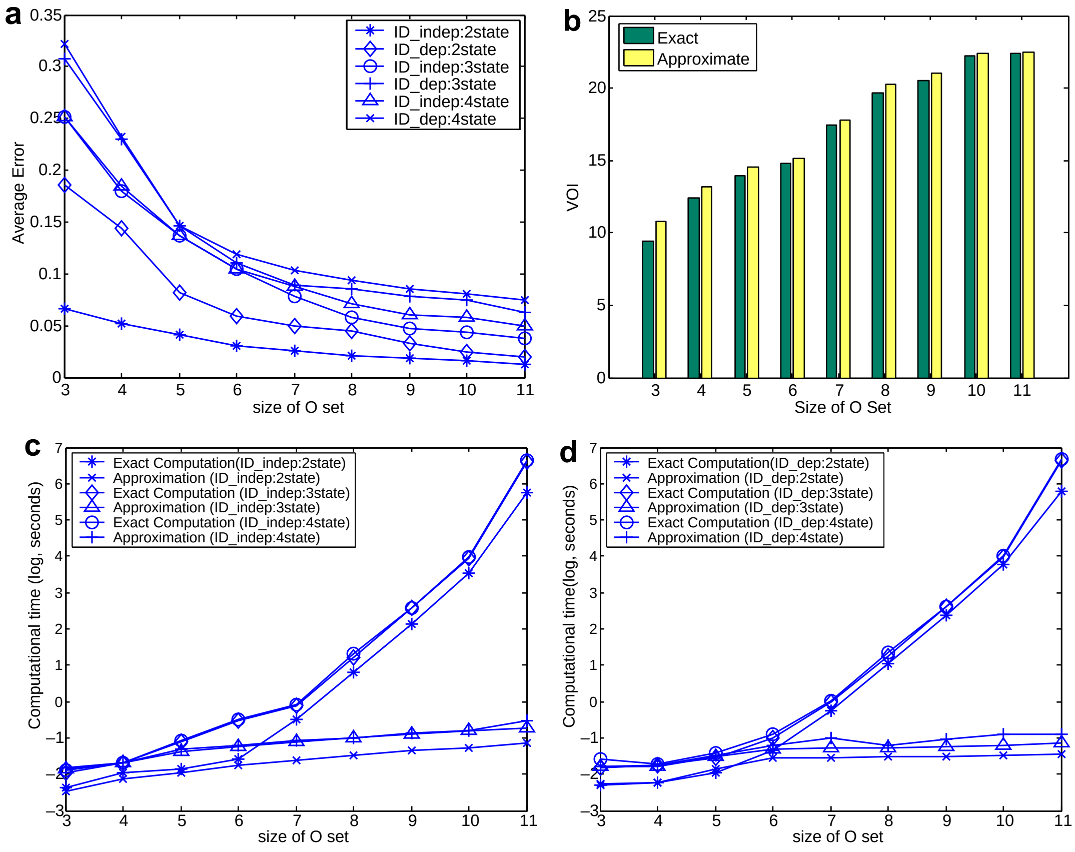


Fig. 9. Results from the four groups of 1500 test cases: (a) average error rates with the approximation algorithm; (b) VOIt vs. VOI for one test case from *ID_dep: 3-state*; (c) computational time (log(t), unit is second) for the groups of *ID_indep:n-state*, $n = 2, 3, 4$; and (d) computational time (log(t), unit is second) for the groups of *ID_dep:n-state*, $n = 2, 3, 4$.

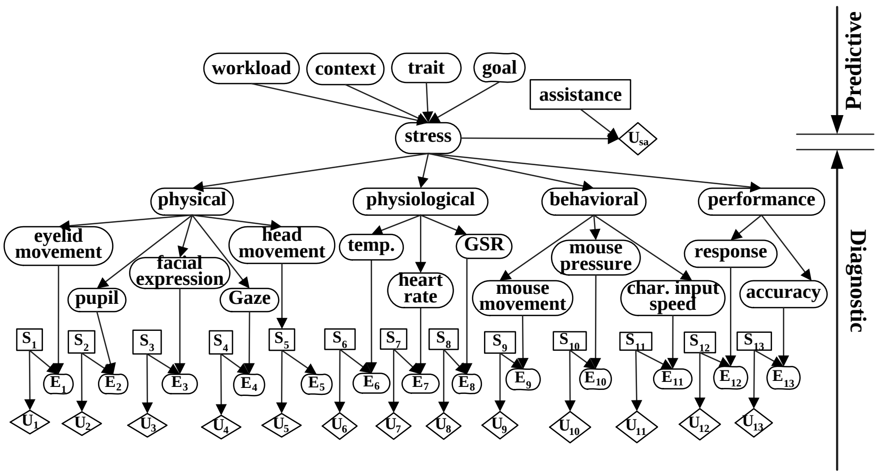


Fig. 10. An influence diagram for recognizing human stress and providing user assistance. Ellipses denote chance nodes, rectangles denote decision nodes, and diamonds denote utility nodes. All the chance nodes have three states.

he/she is pursuing. This portion is called predictive portion. On the other hand, the lower portion of the diagram, from the “stress” node to the leaf nodes, depicts the observable features that reveal stress. These features include the quantifiable measures on the user physical appearance, physiology, behaviors, and performance. This portion is called diagnostic portion.

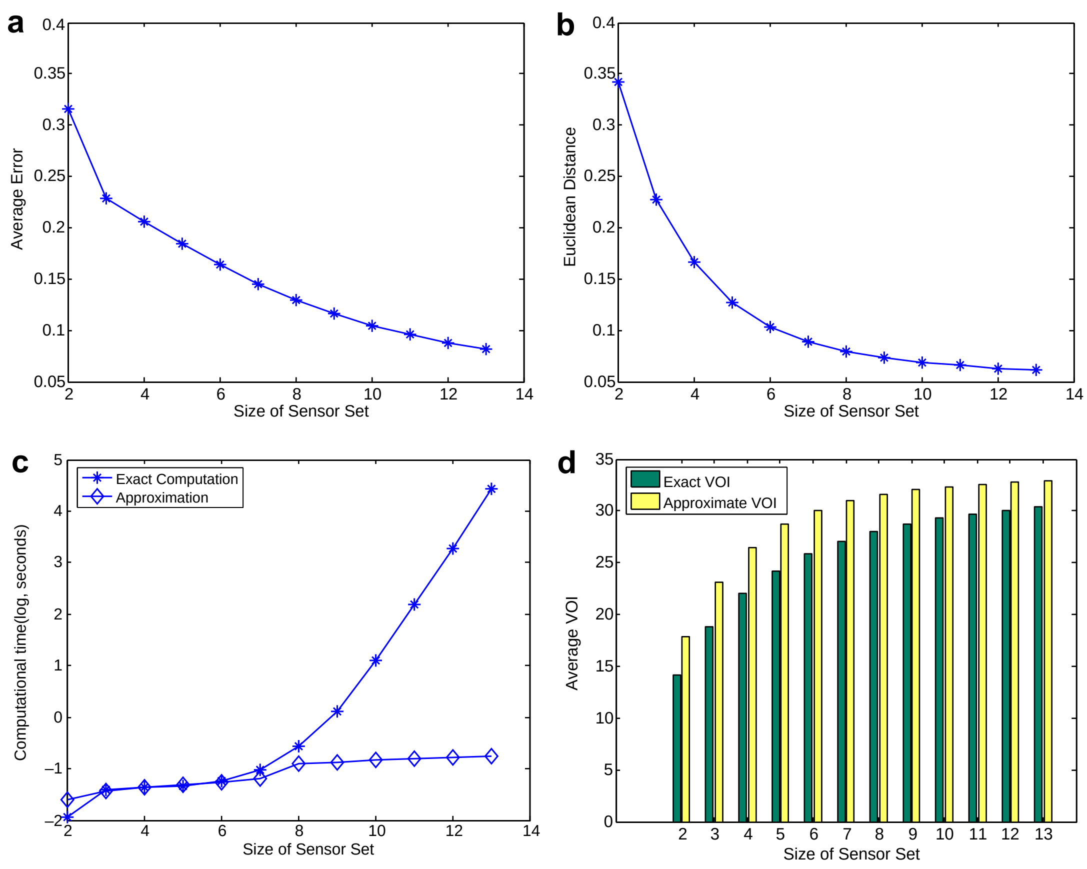


Fig. 11. Results for the stress modeling: (a) average errors with the approximation algorithm; (b) Euclidean distance between the true and approximated $\sum_{o\in o_{d_k}} p(o|\theta_i)$; (c) computational time ($\log(t)$, unit is second); (d) true VOI vs. approximated VOI.

The hybrid structure enables the ID to combine the predictive factors and observable evidence in user stress inference. For more detail please refer to [19].

To provide timely and appropriate assistance to relieve stress, two types of decision nodes are embedded in the model to achieve this goal. The first type is the assistance node associated with the stress node, which includes three types of assistance that have different degrees of impact and intrusiveness to a user. Another type of decision nodes is the sensing action node (S_i node in Fig. 10). It decides whether to activate a sensor for collecting evidence or not. Through the ID, we decide the sensing actions and the assistance action sequentially. In order to first determine the sensing actions (which sensors should be turned on), VOI is computed for a set S consisting of S_i. Using the notations defined before, we have $\mathrm{VOI}(S) = \mathrm{VOI}(E) - \sum_{S_i\in S} u_i(S_i)$, where E is the set of observations corresponding to S and $\mathrm{VOI}(E) = EU(E) - EU(\bar{E})$.

Fig. 11 shows the experimental results for the stress model. We enumerate all the possible combinations of sensors and then compute the value-of-information for each combination. Chart (a) illustrates the average VOI errors for different sensor sets with the same size. And Chart (b) displays the Euclidean distance between the true and estimated probabilities $\sum_{o\in o_{d_k}} p(o|\theta_i)$ (Eq. (26)). Similarly to the simulation experiments, the error decreases as the size of O set increases, and the computational time increases almost linearly in the approximation algorithm.

6. Conclusions and future work

As a concept commonly used in influence diagrams, VOI is widely used as a criterion to rate the usefulness of various information sources, and to decide whether pieces of evidence are worth acquiring before actually using the information sources. Due to the exponential time complexity of computing non-myopic VOI for multiple information sources, most researchers focus on the myopic VOI, which requires the assumptions (“No competition” and “One-step horizon”) that may not meet the requirements of real-world applications.

We thus proposed an algorithm to approximately compute non-myopic VOI efficiently by utilizing the central-limit theorem. Although it is motivated by the method of [11], it overcomes the limitations of their method, and works for more general cases, specifically, no binary-state assumption for all the nodes and no conditional-independence assumption for the

Table 5
The proposed algorithm vs. the algorithm in [11]

Our algorithm	Heckerman's algorithm
Hypothesis node (Θ) can be multiple states	Θ has to be binary
Decision node (D) can have multiple rules	D has to be binary
Information sources nodes (Os) can be dependent from each other	Os have to be conditionally independent from each other

information sources. Table 5 compares our method with the method in [11]. Due to the benefits of our method, it can be applied to a much broader field. The experiments demonstrate that the proposed algorithm can approximate the true non-myopic VOI well, even with a small number of observations. The efficiency of the algorithm makes it a feasible solution in various applications when efficiently evaluating a lot of information sources is necessary.

Nevertheless, the proposed algorithm focuses on the influence diagrams with one decision node under certain assumptions. For example, currently, we assume the hypothesis node Θ and the decision node d are independent. If D and Θ are dependent, but conditionally independent given the observation set O, Eqs. (5) and (6) will not be affected, so our algorithm can still apply. However, if D and Θ are dependent given O, it may be difficult to directly apply our algorithm. Another scenario is that when there are more than one hypothesis node and/or utility nodes. One possible solution is to group all these hypotheses nodes into one. We would like to study these issues in the future.

Appendix

Proposition 1. *Let* $r_{jk} = \frac{u_{2j}-u_{2k}}{u_{1k}-u_{1j}+u_{2j}-u_{2k}}$, $p^*_{kl} = \max_{k<j\leqslant q} r_{jk}$, *and* $p^*_{ku} = \min_{1\leqslant j<k} r_{jk}$, *then* d_k *is the optimal action if and only if* $p^*_{kl} \leqslant p(\theta_1) \leqslant p^*_{ku}$.

Proof of Proposition 1. $\Rightarrow$ In this direction, we prove that if d_k is the optimal action, $p(\theta_1) \geqslant \max_{k<j\leqslant q} r_{jk}$ and $p(\theta_1) \leqslant \min_{1\leqslant j<k} r_{jk}$.

If d_k is the optimal action, $EU(d_k)$ must be larger than or equal to the expected utility of any other action. Based on the definition, the expected utility of taking the action d_k is $EU(d_k) = p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k}$, where $u_{1k} = u(\theta_1, d_k)$, and $u_{2k} = u(\theta_2, d_k)$. Therefore, we get the equations as follows:

$$EU(d_k) \geqslant EU(d_j) \quad \forall j, j \neq k, \tag{27}$$

$$\Rightarrow p(\theta_1) * u_{1k} + p(\theta_2) * u_{2k} \geqslant p(\theta_1) * u_{1j} + p(\theta_2) * u_{2j}, \tag{28}$$

$$\Rightarrow p(\theta_1) \geqslant \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}} = r_{jk} \quad \text{if } j > k, \tag{29}$$

$$p(\theta_1) \leqslant \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}} = r_{jk} \quad \text{if } j < k. \tag{30}$$

Thus, based on the above equations, $p(\theta_1) \geqslant \max_{k<j\leqslant q} r_{jk}$ and $p(\theta_1) \leqslant \min_{1\leqslant j<k} r_{jk}$.

$\Leftarrow$ In this direction, we prove that if $p(\theta_1) \geqslant \max_{k<j\leqslant q} r_{jk}$ and $p(\theta_1) \leqslant \min_{1\leqslant j<k} r_{jk}$, then d_k is the optimal action.

If $p(\theta_1) \geqslant \max_{k<j\leqslant q} r_{jk}\ \forall j, k < j \leqslant q$, we get

$$p(\theta_1) \geqslant r_{jk} = \frac{u_{2j} - u_{2k}}{u_{1k} - u_{1j} + u_{2j} - u_{2k}}, \tag{31}$$

$$\Rightarrow p(\theta_1)(u_{1k} - u_{1j} + u_{2j} - u_{2k}) \geqslant u_{2j} - u_{2k}, \tag{32}$$

$$\Rightarrow p(\theta_1) * u_{1k} + (1 - p(\theta_1)) * u_{2k} \geqslant p(\theta_1) * u_{1j} + (1 - p(\theta_1)) * u_{2j}, \tag{33}$$

$$\Rightarrow EU(d_k) \geqslant EU(d_j). \tag{34}$$

Similarly, for $\forall j, 1 \leqslant j < k$, we can get $EU(d_k) \geqslant EU(d_j)$. Therefore, d_k has the maximal expected utility and thus is the optimal decision.

Proposition 2. $\sum_{o\in o_{d_k}} p(o) = p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku})$.

Proof of Proposition 2. Based on Proposition 1, d_k is the optimal decision if and only if the value of $p(\theta_1)$ is between p^*_{kl} and p^*_{ku}. Therefore, given an instantiation o, the probability that d_k is the optimal decision is equal to the probability that $p(\theta_1|o)$ is between p^*_{kl} and p^*_{ku}, i.e., $p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku})$.

On the other hand, we know that o_{d_k} is a subset of instantiations, each of which corresponds to the optimal action d_k. Therefore, as long as o belongs to the set of o_{d_k}, d_k will be the optimal decision. In other words, the probability of d_k being the optimal decision is the sum of the probability of each $o \in o_{d_k}$, which is $\sum_{o\in o_{d_k}} p(o)$. Therefore, $\sum_{o\in o_{d_k}} p(o) = p(p^*_{kl} \leqslant p(\theta_1|o) \leqslant p^*_{ku})$.

References

[1] N. Balakrishnan, W.W.S. Chen, Handbook of Tables for Order Statistics from Log Normal Distributions with Applications, Kluwer, Amsterdam, Netherlands, 1999.
[2] G. Casella, R. Berger, Statistical Inference, Brooks/Cole, 1990. pp. 45–46 (Chapter 2).
[3] E. Cohen, N. Megiddo, Improved algorithms for linear inequalities with two variables per inequality, SIAM Journal on Computing 23 (6) (1994) 1313–1347.
[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press and McGraw-Hill, 2002.
[5] E. Crow, K. Shimizu, Lognormal Distributions: Theory and Applications, Dekker, New York, 1988.
[6] M. Diehl, Y. Haimes, Influence diagrams with multiple objectives and tradeoff analysis, IEEE Transactions on Systems, Man and Cybernetics, Part A 34 (3) (2004) 293–304.
[7] S. Dittmer, F. Jensen, Myopic value of information in influence diagrams, Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (1997) 142–149.
[8] K.J. Ezawa, Evidence propagation and value of evidence on influence diagrams, Operations Research 46 (1) (1998) 73–83.
[9] W. Feller, An Introduction to Probability Theory and Its Applications, third ed., vol. 2, Wiley, New York, 1971.
[10] J. Hammersley, Monte Carlo methods for solving multivariable problems, Annals of the New York Academy Sciences 86 (1960) 844–874.
[11] D. Heckerman, E. Horvitz, B. Middleton, An approximate nonmyopic computation for value of information, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (3) (1993) 292–298.
[12] R. Howard, Value of information lotteries, IEEE Transactions of Systems Science and Cybernetics 3 (1) (1967) 54–60.
[13] R. Howard, J. Matheson, Influence diagrams, Readings on the Principles and Applications of Decision Analysis 2 (1981) 721–762.
[14] F. Jensen, Bayesian Networks and Decision Graphs, Springer-Verlag, New York, 2001.
[15] F. Jensen, F.V. Jensen, S. Dittmer, From influence diagrams to junction trees, (1994) 367–374.
[16] M. Kalos, P. Whitlock, Monte Carlo Methods, Wiley, New York, 1986.
[17] K.B. Korb, A.E. Nicholson, Bayesian Artificial Intelligence, Chapman and Hall/CRC, 2003.
[18] W. Liao, Q. Ji, Efficient active fusion for decision-making via VOI approximation. Twenty-first National Conference on Artificial Intelligence (AAAI), 2006.
[19] W. Liao, W. Zhang, Z. Zhu, Q. Ji, A decision theoretic model for stress recognition and user assistance, Twentieth National Conference on Artificial Intelligence (AAAI) (2005) 529–534.
[20] Mathematica, 2006. <http://www.wolfram.com/products/mathematica/index.html>.
[21] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann Publishers, 1988.
[22] K.L. Poh, E. Horvitz, A graph–theoretic analysis of information value, Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96) (1996) 427–435.
[23] H. Raiffa, Decision Analysis, Addison-Wesley, 1968.
[24] R. Shachter, Evaluating influence diagrams, Operations Research 34 (6) (1986) 871–882.
[25] R. Shachter, Efficient value of information computation, Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99) (1999) 594–601.
[26] R. Shachter, P.M. Ndilikilikesha, Using potential influence diagrams for probabilistic inference and decision making, Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence (UAI-93) (1993) 383–390.
[27] F. Yokota, K.M. Thompson, Value of information analysis in environmental health risk management decisions: past, present, and future, Risk Analysis 24 (2004) 635–650.
[28] N.L. Zhang, R. Qi, D. Poole, Incremental computation of the value of perfect information in stepwise-decomposable influence diagrams, Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence (1993) 400–410.

Efficient Sensor Selection for Active Information Fusion

Yongmian Zhang, *Member, IEEE*, and Qiang Ji, *Senior Member, IEEE*

***Abstract*—In our previous paper, we formalized an active information fusion framework based on dynamic Bayesian networks to provide active information fusion. This paper focuses on a central issue of active information fusion, i.e., the efficient identification of a subset of sensors that are most decision relevant and cost effective. Determining the most informative and cost-effective sensors requires an evaluation of all the possible subsets of sensors, which is computationally intractable, particularly when information-theoretic criterion such as mutual information is used. To overcome this challenge, we propose a new quantitative measure for sensor synergy based on which a sensor synergy graph is constructed. Using the sensor synergy graph, we first introduce an alternative measure to multisensor mutual information for characterizing the sensor information gain. We then propose an approximated nonmyopic sensor selection method that can efficiently and near-optimally select a subset of sensors for active fusion. The simulation study demonstrates both the performance and the efficiency of the proposed sensor selection method.**

***Index Terms*—Active information fusion, Bayesian networks (BNs), sensor selection, situation awareness.**

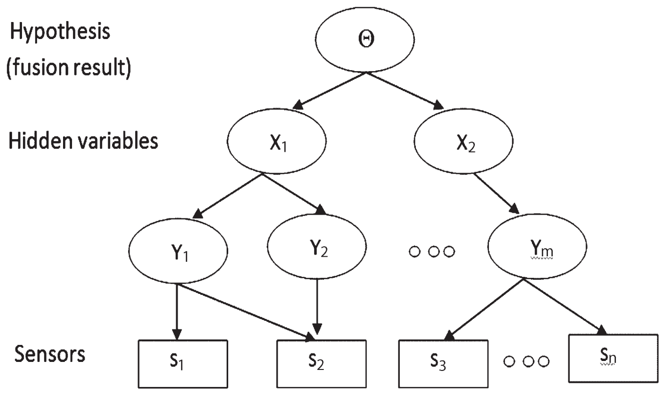


Fig. 1. BN is used for active information fusion, where Θ and S_i are hypothesis and sensors, respectively. X_i and Y_i are the intermediate variables, and they are needed to model the relationships among sensors and the hypothesis. Sensor fusion is accomplished through probabilistic inference given the sensory measurements.

I. Introduction

INFORMATION fusion is playing an increasingly important role in improving the performance of sensory systems for various applications, including situation assessment, enemy intent understanding and prediction, and threat assessment. As sensors become ubiquitous, persistent, and pervasive, and coupled with the ever increasing demand for less time and fewer resources, it becomes critically important to perform selective fusion so that decision can be made in a timely and efficient manner. The need for sensor selection is further demonstrated by the availability of an increasingly large volume of sensory data and by the variability of sensor reliability over time and over location. It is important to select the sensors not only to reduce the amount of data to integrate but also to improve fusion accuracy by selecting the most reliable sensors for a certain location at a certain time, by selecting complementary sensors, and by reducing sensor redundancy. Active fusion serves these purposes well. Active fusion extends the paradigm of information fusion by being not only concerned with the methodology of how to combine information but also concerned with the fusion efficiency, timeliness, and accuracy. Active fusion can be defined as the process of combining data with a control mechanism that dynamically selects a subset of sensors to minimize uncertainty in situation assessment and to maximize the overall expected utility in decision making.

In our previous work [1], we formalized an information fusion framework based on Bayesian networks (BNs) to provide active and sufficing information fusion. BNs are used to model a number of uncertain events, their spatial relationships, and the sensor measurements. Given the sensory measurements, information fusion is performed through probabilistic inference using the BN. This can be accomplished through bottom-up belief propagation, as illustrated in Fig. 1. Our previous work, however, did not address the core issue in active fusion, i.e., efficient sensor selection. This is the focus of this paper.

Based on information theory [2], the more sensors[1] we use, the more information we can obtain. However, every act of information gathering incurs cost. Sensor costs may include physical costs, computational costs, maintenance costs, and human costs (e.g., risk). Many applications are often constrained by limited time and resources. An essential issue for active information fusion is to select a subset of the most

Manuscript received May 20, 2008; revised October 25, 2008, December 19, 2008, and March 11, 2009. This work was supported in part by the Air Force Office of Scientific Research under Grant F49620-03-0160. This paper was recommended by Associate Editor R. Lynch.

The authors are with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA (e-mail: zhangy@ecse.rpi.edu; qji@ecse.rpi.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMCB.2009.2021272

[1]For generality, sensors could refer to any devices/means of acquiring information. For example, they may be electromagnetic or acoustic devices or they could also be direct observations of the world through reconnaissance and intelligence gathering activities.

1083-4419/$26.00 © 2009 IEEE

synergetic sensors, which can maximally reduce the uncertainty about the events of interest with minimum costs. Dynamically determining the best set of sensors, given the uncertainty about the state of the world, requires to enumerate all the possible subsets of sensors, which is computationally intractable and practically infeasible. This computational difficulty is twofold. First, the computation of a sensor selection criterion such as mutual information is exponential with respect to the number of sensors. Second, searching for an optimal subset of sensors is also NP-hard, since the sensor space exponentially increases with the number of sensors. To address this computational difficulty, a common practice is to use myopic analysis, which assumes that only one observation will be available at a time, even when there is an opportunity to make a set of observations [3]–[6]. There is a vast literature on the problem of single optimal sensor selection [7]–[9]. However, the myopic approach cannot guarantee to obtain the best evidences that most effectively reduce uncertainty and cost. To effectively reduce uncertainty and cost, one should use nonmyopic selection, which simultaneously considers several observations before making a decision. The most common nonmyopic method is the greedy approach. While efficient, it cannot guarantee optimality with the selected sensors. Other works try to overcome the limitations with the greedy approach, yet with their own strong assumptions. In [10], Heckerman *et al.* presented an approximate nonmyopic approach based on the central-limit theorem in an influence diagram (ID) for efficiently computing the value of information. Their method, however, assumes that the sensors are conditionally independent of each other, given the decision variable, and that the decision variable is binary. Krause and Guestrin [11], [12] presented a randomized approximation algorithm for selecting a near-optimal subset of observations for graphical models. Under the assumptions that the sensors are conditionally independent given the decision variables, the information gain is then guaranteed to be a submodular function, and the theory of submodular functions can then be applied to achieve a near-optimal solution in selecting a subset of observations using a greedy approach. Recently, Liao and Ji [13] have presented an approximation algorithm for the nonmyopic computation of the value of information in an ID. Their method extends the approach in [10] without requiring the sensors being conditionally independent of each other and the decision node being binary.

This paper takes another avenue of approach to efficiently select a subset of near-optimal sensors without the strong sensor independence assumptions, as made in [10] and [12]. Specifically, we first introduce a new quantitative measure of sensor synergy based on mutual information. Based on the synergy measure, we then introduce a method to efficiently compute the least upper bound (LUB) of mutual information for a set of sensors. Experiments show that the LUB closely approximates the mutual information in value, as shown in Figs. 5 and 6. Hence, the computational difficulty with computing the exact nonmyopic mutual information can, therefore, be circumvented by computing its LUB instead. In addition, the synergy measure can also be used to prune the sensor space, which, therefore, reduces the search time for the best sensor set. A summary of the mentioned work may be found in [14].

II. Problem Formulation

The problem of sensor selection for active fusion can be stated as follows: Assume that there are m sensors S_i, $i = 1, \dots, m$, available that provide measurements of the world. Let Θ be a set of hypothesis θ_k of the world situation $k = 1, \dots, K$. Let $\mathbf{S} = \{S_1, \dots, S_n\}$ be a subset of n sensors selected at time t, where $n \in \{1, \dots, m\}$. Let $C(\mathbf{S})$ be the cost to use the set of sensors $\mathbf{S}$. The objective of sensor selection at time $t+1$ is to select a subset of sensor $\mathbf{S}^*$ to achieve the maximal utility, i.e.,

$$\mathbf{S}^* = \arg\max_{\mathbf{S} \in \mathcal{S}} U(u_1, u_2) \tag{1}$$

where u_1 and u_2 denote information gain (i.e., the mutual information) and the sensor usage cost saving, respectively, $\mathcal{S}$ represents all the possible subsets of sensors, and $U(u_1, u_2)$ is a utility function. Here, we use $u_2 = 1 - C(\mathbf{S})$ to convert the sensor usage cost to the corresponding cost saving, which makes u_1 and u_2 qualitatively equivalent. For simplicity, in this paper, we assume that the cost is the same for all sensors. Hence, we can ignore u_2.

The major difficulty of using (1) for sensor selection is to efficiently compute the information gain u_1. From information theory, the entropy of hypothesis Θ given a sensor S_i measures how much uncertainty exists in Θ given S_i, i.e.,

$$H(\Theta \mid S_i) = -\sum_{s_i} \sum_{\Theta} P(\theta, s_i) \log P(\theta \mid s_i) \tag{2}$$

where s_i denotes a reading of sensor S_i.[2] Subtracting $H(\Theta \mid S_i)$ from the original uncertainty in Θ without S_i, i.e., $H(\Theta)$, yields the expected amount of information about Θ that S_i is capable of providing

$$\begin{aligned} I(\Theta; S_i) &= H(\Theta) - H(\Theta \mid S_i) \\ &= -\sum_{\Theta} P(\theta) \log P(\theta) \\ &\quad + \sum_{S_i} \left\{ P(s_i) \sum_{\Theta} P(\theta \mid s_i) \log P(\theta \mid s_i) \right\} \\ &= \sum_{\Theta} \sum_{S_i} P(\theta, s_i) \log \frac{P(\theta \mid s_i)}{P(\theta)} \end{aligned} \tag{3}$$

where $I(\Theta; S_i)$ is referred to as the mutual information, which characterizes the expected total uncertainty-reducing potential of Θ due to S_i. The mutual information for a sensor set $\mathbf{S} = \{S_1, \dots, S_n\}$ can be obtained by

$$\begin{aligned} &I(\Theta; \mathbf{S}) \\ &= H(\Theta) - H(\Theta \mid \mathbf{S}) \\ &= -\sum_{\Theta} P(\theta) \log P(\theta) \\ &\quad + \sum_{\Theta} \sum_{S_1} \cdots \sum_{S_n} \{ P(\theta, s_1, \dots, s_n) \log P(\theta \mid s_1, \dots, s_n) \} \\ &= \sum_{\Theta} \sum_{S_1} \cdots \sum_{S_n} \left\{ P(\theta, s_1, \dots, s_n) \log \frac{P(\theta \mid s_1, \dots, s_n)}{P(\theta)} \right\} \end{aligned} \tag{4}$$

[2]Without loss of generality, here we assume discrete sensor measurement. The theories can be straightforwardly extended to continuous sensor measurements.

where $P(\theta, s_1, \ldots, s_n)$ and $P(\theta \,|\, s_1, \ldots, s_n)$ at time t can directly be obtained through BN inference. The mutual information in (4) provides a sensor selection criterion in terms of the uncertainty reduction potential, i.e., mutual information.

It is clear from (4) that when the number of sensors in $\mathbf{S}$ is large or when the number of states for each sensor is large, it becomes computationally impractical to simply implement this information-theoretic criterion, because it generally requires time exponential in the number of summations to exactly compute the mutual information. The remainder of this paper addresses this computational difficulty.

III. Approximation Algorithm

In this section, we give a graph-theoretic definition of sensor synergy. We then present the theorems on which our algorithm is based.

A. Sensor Synergy in Information Gain

Throughout this section, it is assumed that we have obtained $I(\Theta; S_i, S_j)$ and $I(\Theta; S_i)$, i.e., the mutual information of all pairs of sensors and individual sensors with respect to Θ, respectively. We will introduce an efficient method to obtain all $I(\Theta; S_i, S_j)$ in Section III-C. We first define a synergy coefficient to characterize the synergy between two sensors, and then extend this definition to multiple sensors.

Definition 1 (Synergy Coefficient): A measure of the expected synergetic potential between two sensors S_i and S_j in reducing the uncertainty of hypothesis Θ is defined as

$$r_{ij} = \frac{I(\Theta; S_i, S_j) - \max(I\,(\Theta; S_i), I(\Theta; S_j))}{H(\Theta)}. \tag{5}$$

The denominator $H(\Theta)$ in (5) is to restrict r_{ij} to the interval [0, 1]. It can easily be proved that $r_{ij} \geq 0$ based on the "information never hurts" principle [2], i.e., $I(\Theta; S_i, S_j) \geq I(\Theta; S_i)$, and $I(\Theta; S_i, S_j) \geq I(\Theta; S_j)$. This follows that S_i and S_j taken together are always more informative than when they are taken alone. The larger r_{ij} is, the more synergetic the sensors S_i and S_j are. Obviously, $r(\cdot,\cdot)$ is symmetrical in S_i and S_j, and $r_{ij} = 0$ if $i = j$.

Definition 2 (Synergy Matrix): Let a sensor set be $\mathbf{S} = \{S_1, \ldots, S_n\}$. The sensor synergy matrix is an $n \times n$ matrix defined as

$$R = \begin{bmatrix} 0 & r_{12} & \cdots & r_{1n} \\ r_{21} & 0 & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ r_{n1} & r_{n2} & \cdots & 0 \end{bmatrix}. \tag{6}$$

R is an information measure of synergy among sensors that is based on pairwise sensor synergy. With a synergy matrix, we naturally define its graphical representation.

Definition 3 (Synergy Graph): Given a sensor synergy matrix, a graph $G = (\mathbf{S}, \mathbf{E})$, where $\mathbf{S}$'s are the nodes representing the set of available sensors, and $\mathbf{E}$'s are the links representing the set of pairwise synergetic links weighted by synergy coefficients r_{ij}, is a sensor synergy graph.

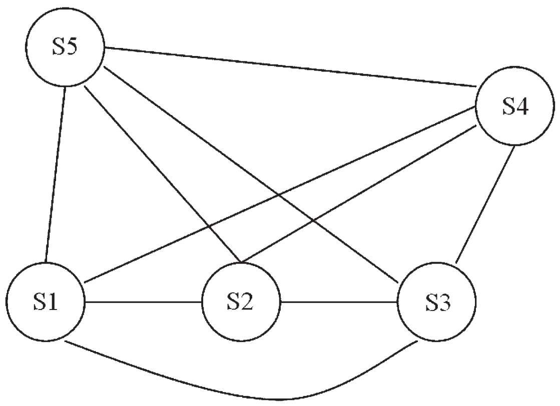


Fig. 2. Example of synergy graph with five sensors.

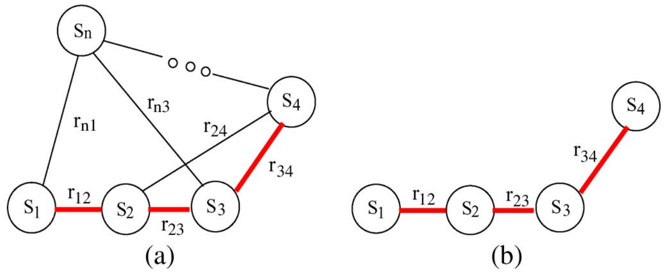


Fig. 3. (a) Synergy chain $\{S_1, S_2, S_3, S_4\}$ (highlighted) on a pruned synergy graph. (b) Corresponding MSC.

We use the synergy graph to graphically represent the synergy among multiple sensors. By definition of the synergy, G is a complete graph, i.e., there is a link between any two nodes in the graph. Fig. 2 gives an example synergy graph consisting of five sensors.

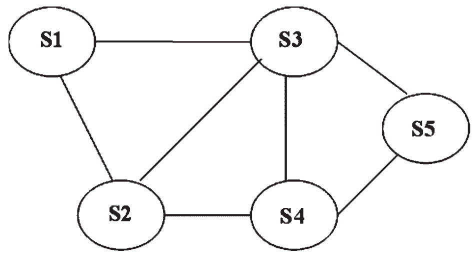
Definition 4 (Pruned Synergy Graph): A pruned synergy graph is created from a synergy graph after removing some links. A pruned synergy is, therefore, not a complete graph.

Fig. 3 shows an example of a pruned synergy graph. To further exploit the theoretical properties of mutual information $I(\Theta; \mathbf{S})$ for a set of sensors, we give the following definitions.

Definition 5 (Synergy Chain): Given a pruned synergy graph G, if all the sensors in a subset on G are serially linked, then this subset of sensors is referred to as a sensor synergy chain. Note that while the sensors in a set $\mathbf{S}$ are generally order independent, the sensors in a synergy chain are order dependent and sequentially ordered.

Definition 6 (MSC): Given a synergy chain with n sensors, for all $i = 1, \ldots, n-1$, if $p(S_{i+1} \,|\, S_1, S_2, \ldots, S_i) = p(S_{i+1} \,|\, S_i)$, then the chain that describes the synergetic relationship among $\{S_1, \ldots, S_n\}$ is a Markov synergy chain (MSC). An MSC is also ordered.

Fig. 3 graphically shows the above definitions about the synergy chain in a pruned synergy graph. The MSC represents an ideal synergetic relationship among sensors. The MSC rarely exists in practice, but this does not prevent us from using it as a basis for the graph-theoretic analysis of synergy among sensors. In fact, as to be shown later, the concept of MSC is used to define the upper bound for the mutual information of a set of sensors. With the above definitions, we give the following two theorems.



Sensor set: 1, 2, 3, 4
Corresponding Markov Synergy Chains:
1234; 1243; 1324; 1342; 2134 2431; 3421; 312 4; 4312; 4231; 4213

Fig. 4. Illustration of a set of possible MSCs for a set unordered four sensors in a pruned synergy graph.

Theorem 1 (MSC Rule): Given an MSC with a set of ordered sensors $\mathbf{S} = \{S_1, \ldots, S_n\}$, for any n, the joint mutual information with respect to Θ for sensors on an MSC is

$$I^M(\Theta; S_1, \ldots, S_n) = I(\Theta; S_1) + \sum_{i=1}^{n-1} \left(I(\Theta; S_i, S_{i+1}) - I(\Theta; S_i) \right). \quad (7)$$

The proof of this theorem can be found in Appendix A. We want to make note that the mutual information for an MSC is sensor-order dependent due to the pairwise synergy definition. The significance of Theorem 1 is that it allows us to efficiently compute the joint mutual information for n $(n > 2)$ ordered sensors as a sum of mutual information of only singleton and pairwise sensors if the set of sensors forms an MSC. In contrast to (4), the computational cost of (7) is dramatically reduced. Although (7) is particularly for an MSC, the theorem above has some useful properties that can be used for the solution of our sensor selection problem.

Theorem 2 (Synergy Upper Bound): For a set of unordered sensors $\mathbf{S} = \{S_1, \ldots, S_n\}$, its mutual information is upper bounded by the mutual information of the corresponding MSC, i.e.,

$$I(\Theta; S_1, \ldots, S_n) \leq I^M(\Theta; S_1, \ldots, S_n). \quad (8)$$

The proof of this theorem is provided in Appendix B. Please note that while $I(\Theta; S_1, \ldots, S_n)$ is sensor-order independent, $I^M(\Theta; S_1, \ldots, S_n)$ is sensor-order dependent. As a result, depending on the order of sensors in $\mathbf{S}$, different MSCs may be produced. Let

$$I^M_{\min} = \arg\min_{\mathcal{S}} \left(I^M(\Theta; \mathcal{S}) \right)$$
$$I^M_{\max} = \arg\max_{\mathcal{S}} \left(I^M(\Theta; \mathcal{S}) \right) \quad (9)$$

where $\mathcal{S}$ denotes all the possible orders of a sensor set $\{S_1, \ldots, S_n\}$. $I^M_{\min}$ is referred to as the LUB of $I(\Theta; S_1, \ldots, S_n)$, and $I^M_{\max}$ is referred to as the greatest upper bound (GUB) of $I(\Theta; S_1, \ldots, S_n)$. For example, in Fig. 4, the sensor set $\mathbf{S} = \{S_1, S_2, S_3, S_4\}$ has multiple MSCs, as given in this figure, and there exist a LUB and a GUB of $I(\Theta; \mathbf{S})$.

We are particularly interested in the LUB of $I(\Theta; \mathbf{S})$ due to two reasons. First, it can be seen from Figs. 5 and 6 that the LUBs of $I(\Theta; \mathbf{S})$ closely follow the trend of $I(\Theta; \mathbf{S})$ in the entire space of sensor subsets. Second, the exact value of $I(\Theta; \mathbf{S})$ and its LUB are quantitatively very close in value. Thus, $I^M_{\min}(\Theta; \mathbf{S})$ provides a substitute measure for $I(\Theta; \mathbf{S})$

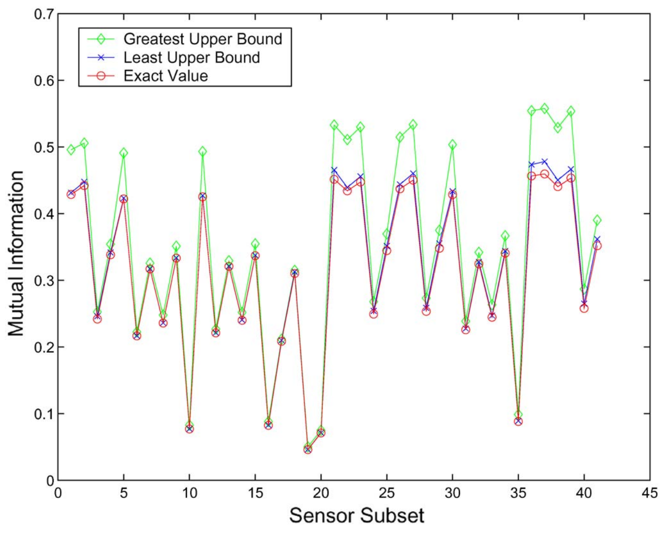


Fig. 5. Bound of mutual information $I(\Theta, S)$ and its exact value from a six-sensor BN model. The X-axis represents the indexes of 41 sensor subsets. Labels 1–20 are the indexes of the three-sensor subsets; Labels 21–35 are the indexes of the four-sensor subsets; and Labels 36–41 are the indexes of the five-sensor subsets.

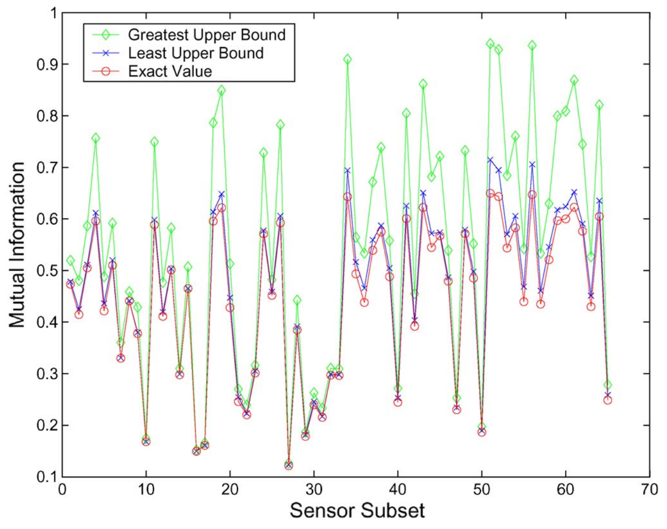


Fig. 6. Bound of mutual information $I(\Theta, S)$ and its exact value from a ten-sensor BN model. The X-axis represents the indexes of sensor subsets. For clarity, the figure only shows 66 subsets out of 627. Labels 1–18 are the indexes of the five-sensor subsets; Labels 19–34 are the indexes of the six-sensor subsets; Labels 35–51 are the indexes of the seven-sensor subsets; and Labels 52–66 are the indexes of the eight-sensor subsets.

that can be used to evaluate an optimal sensor subset. Importantly, the LUBs of $I(\Theta; \mathbf{S})$ can simply be written as the sum of the mutual information of only pairwise sensors and singleton sensors, as shown in (7), hence, with relatively very low computational cost. Therefore, the computational difficulty in exactly computing the higher-order mutual information can be circumvented by only computing the LUBs of the mutual information. This is the central strategy of our approach.

B. Pruning Synergy Graph

The synergy graph is a completely connected network due to the weights of synergy graph $r_{ij} \geq 0$. Some sensors are highly synergetic, whereas others are not. Intuitively, sensors that

TABLE I
EXAMPLE OF SYNERGY COEFFICIENT WITHOUT PRUNING

r_{ij}	1	2	3	4	5	6	7	8	9	10
1	0	0.0004	0.0034	0.0034	0.0029	0.0023	0.0035	0.0035	0.0031	0.0017
2		0	0.0004	0.0004	0.0004	0.0003	0.0004	0.0004	0.0004	0.0002
3			0	0.0215	0.0196	0.0156	0.0099	0.0122	0.0212	0.0113
4				0	0.0184	0.0147	0.0099	0.0122	0.0198	0.0105
5					0	0.0930	0.0083	0.0104	0.0650	0.0659
6						0	0.0066	0.0082	0.0517	0.1329
7							0	0.0100	0.0091	0.0050
8								0	0.0112	0.0060
9									0	0.0388
10										0

cause a very small reduction in uncertainty of hypotheses are those that give us the least additional information beyond what we would obtain from other sensors. In such cases, r_{ij} is very small. We prune the sensor synergy graph so that many weak sensor combinations are eliminated while preserving the most promising ones. This can significantly reduce the search space in identifying the optimal sensor subset. We prune the synergy matrix (the weights of the synergy graph) in (6) by using

$$r_{ij} = \begin{cases} 1, & r_{ij} > \tau \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where τ is a pruning threshold. The selection of an appropriate threshold τ is problem dependent. We want to note that although there is no theoretical basis to determine a good pruning threshold, our empirical tests, however, show that using the arithmetic average of r_{ij} as the pruning threshold can preserve most of the strong synergetic connections in the graph while eliminating weak links. After pruning, a fully connected synergy graph then becomes a sparse graph. Table I and

$$R = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ & & & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ & & & & 0 & 1 & 0 & 0 & 1 & 1 \\ & & & & & 0 & 0 & 0 & 1 & 1 \\ & & & & & & 0 & 0 & 0 & 0 \\ & & & & & & & 0 & 0 & 0 \\ & & & & & & & & 0 & 1 \\ & & & & & & & & & 0 \end{bmatrix} \tag{11}$$

are examples of the synergy coefficient before and after pruning. Fig. 7 illustrates their corresponding synergy graph from a completely connected network to a sparse graph after pruning.

C. Computing Pairwise Mutual Information

In the above sections, we assumed that we have known the mutual information of the pairwise sensors $I(\Theta; S_i, S_j)$. For n sensors, there are $(n(n-1)/2)$ pairs of sensors. To obtain the mutual information for one pair of sensors, it requires four repetitions of inferences if the sensor state is binary. Therefore, $2n(n-1)$ repetitions of inference are needed for

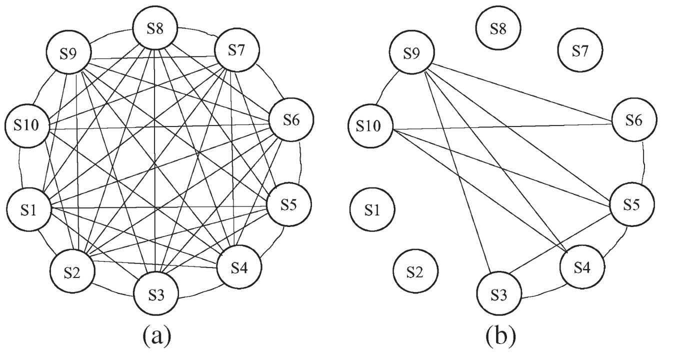


Fig. 7. (a) Completely connected synergy graph and the links are weighted by r_{ij}, as shown in Table I. (b) Pruned synergy graph and its corresponding matrix as shown in (11). The pruning threshold is the average of r_{ij}, and it is 0.0161.

all pairs of sensors. Although this computation is manageable, it still severely limits the performance as n becomes large. Fortunately, there is an efficient way to compute the mutual information for all pairs of sensors [15], [16].

Referring to Fig. 1, the joint probability of hypothesis Θ and pairwise sensors $\{S_i, S_j\}$ may be written as in (12), shown at the bottom of the next page, where $\pi(x)$ represents the parental nodes of node x. From (12), it can be observed that the first factor $P(\Theta)\prod_{k=1}^{K} P(X_k \mid \pi(X_k)) \prod_{m=1}^{M} P(Y_m \mid \pi(Y_m))$ is related to the part of the BN structure that does not include the sensors. The structure is, therefore, fixed, and so are its probabilities. Hence, this term is constant, independent of the pair of sensors used. On the other hand, the second factor $\{\sum_{S_1,S_l,\ldots,S_N, l\neq i\ l\neq j} \prod_{n=1}^{N} P(S_n \mid \pi(S_n))\}$ varies, depending on the pair of sensors selected. Therefore, we do not need to recalculate the unchanged part (the first factor) of (12) at each time. Instead, we only need to compute it once for all pairs of sensors, but use it over time so that the computation of pairwise mutual information can significantly be curtailed. Given $P(\Theta, S_i, S_j)$, it can then be substituted into (4) to compute $I(\Theta; S_i, S_j)$. Details of this method can be found in [15] and [16]. Fig. 8 illustrates the comparative result of time saving in computing (12) for all pairs of sensors by using our method and by directly using two inference algorithms, namely, clique tree propagation (CTP) [17], [18] and variable elimination (VE) [19]. The evaluation is performed on a six-layer BN model with 10, 15, and 20 sensors.

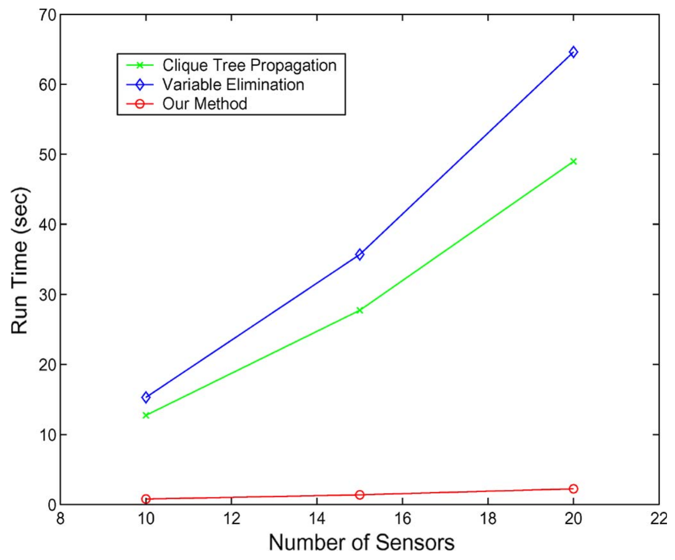


Fig. 8. Comparison of time saving among CTP, VE, and our method in computing (12) for all pairs of sensors. It can be seen that our method can significantly save time.

D. Approximation Algorithm

We are now ready to provide the complete algorithm. Let $\mathbf{S}$ denote the current set of selected sensors, and let $lub(\Theta;\mathbf{S})$ be the LUB of $I(\Theta;\mathbf{S})$. The approximation sensor selection algorithm is given in Table II. Guided by the pruned synergy graph, the algorithm starts with the best pair of sensors identified through an exhaustive search and then searches for the next best sensor. The next best sensor is the one, when added to the current sensor ensemble, that yields the highest utility, which is computed from $lub(\Theta;\mathbf{S})$. This process repeats, with one sensor added to the current sensor ensemble at a time, until the newly added sensor does not yield an improvement in sensor utility. Although the algorithm is greedy, the searching process is guided by a synergy graph so that the selected sensor subset is serially connected. This, therefore, ensures both the quality and the speed of sensor selection.

TABLE II
PSEUDOCODE OF THE APPROXIMATION ALGORITHM TO SELECT A SUBSET OF SENSORS

```
SENSOR-SELECTION($n$)
1  for each $i, j$, compute $I(\Theta;S_i)$ and $I(\Theta;S_i,S_j)$
2  Construct a pruned synergy graph $G$
3  Choose $S_{i^*}$, $S_{j^*}$ such that $I(\Theta;S_i,S_j)$
      is maximized for all $i$ and $j$
4  $\mathbf{S} \leftarrow \{S_{i^*}, S_{j^*}\}$
5  while $|\mathbf{S}| < m$
6     for each $\mathbf{S}'$, where $|\mathbf{S}'| = |\mathbf{S}| + 1$, and $\mathbf{S}'$ is a synergy
         chain on $G$ and $\mathbf{S} \subset \mathbf{S}'$
7        Find all Markov synergy chains of $\mathbf{S}'$
9        $lub(\Theta;\mathbf{S}') \leftarrow \arg\min I(\Theta;\mathbf{S}')$,
            where $I(\Theta;\mathbf{S}')$ is computed by Eq. (7), and
            min is taken over all Markov synergy chains of $\mathbf{S}'$
10    $\mathbf{S}'^* \leftarrow \arg\max lub(\Theta;\mathbf{S}')$
         where max is taken over all $\mathbf{S}'$
11    if $lub(\Theta;\mathbf{S}'^*) > lub(\Theta;\mathbf{S})$
12       $\mathbf{S} \leftarrow \mathbf{S}'^*$
13    else break
14 return $\mathbf{S}$
```

IV. Algorithm Evaluation

Since the main contribution of this paper is the introduction of an alternative measure to mutual information for efficient sensor selection, the experimental evaluation should focus on the effectiveness of this measure for both sensor selection accuracy and efficiency. We want to emphasize that the alternative measure, i.e., the LUB of mutual information, is an approximation of the mutual information only for the purpose of sensor selection. As a result, the quality of this approximation should be evaluated against its performance in sensor selection. For this, we propose to measure how close the sensor selection results using the alternative measure are to those based on mutual information. The closeness between a sensor subset selected using the alternative measure and a sensor subset selected based on mutual information is quantified by the relative difference in mutual information. Based on this criterion, we will experimentally evaluate the proposed method under different BN topologies, different BN model complexities, and different number of sensors.

Given two different criteria (mutual information and its LUB) for measuring sensor gain, sensor selection can be carried out by using different methods. We will perform sensor selection using the following methods: 1) brute-force method; 2) random method, which randomly chooses one sensor at a time to form a sensor ensemble; and 3) the proposed method. These experiments try to demonstrate the following: 1) The proposed LUB criterion suboptimally works for different methods. 2) Given the same sensor selection criterion, the proposed greedy approach outperforms the random sensor selection method.

$$\begin{aligned} P(\Theta,S_i,S_j) &= \sum_{S_1,S_l,\ldots,S_n,l\neq i\ l\neq j} \left\{ P(\Theta)\prod_{k=1}^{K} P\left(X_k \,|\, \pi(X_k)\right) \prod_{m=1}^{M} P\left(Y_m \,|\, \pi(Y_m)\right) \prod_{n=1}^{N} P\left(S_n \,|\, \pi(S_n)\right) \right\} \\ &= P(\Theta)\prod_{k=1}^{K} P\left(X_k \,|\, \pi(X_k)\right) \prod_{m=1}^{M} P\left(Y_m \,|\, \pi(Y_m)\right) \left\{ \sum_{S_1,S_l,\ldots,S_n,l\neq i\ l\neq j} \prod_{n=1}^{N} P\left(S_n \,|\, \pi(S_n)\right) \right\} \end{aligned} \tag{12}$$

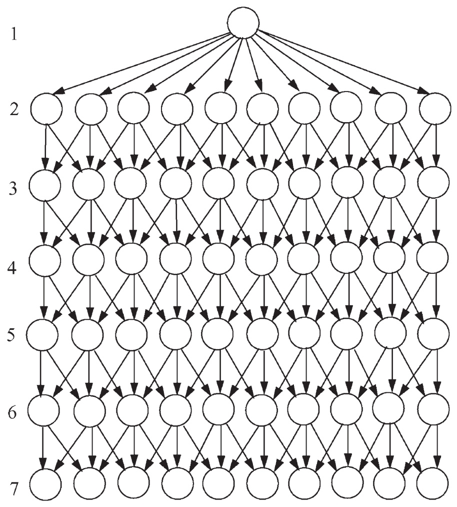


Fig. 9. Generic example of the BN network used for evaluation, where the top layer is for the hypothesis, and the bottom layer is for the sensors. The intermediate layers are arbitrarily and randomly connected.

We first compare the performance of the proposed sensor selection method in Table II with the brute-force method. The brute-force method exhaustively identifies the best sensor subset by the exact mutual information. The study is done by using different numbers of sensors and different BN topologies. Fig. 9 shows a generic example of a BN used for the evaluation.

Due to the exponential time with the brute-force approach, we limit our test models to up to five layers and up to ten sensors or less. The exact number of layers, the connections among nodes in the intermediate layers, and the number of sensors are randomly generated so that ten different BNs with different topologies are generated. For each randomly generated BN topology, its parameters are randomly parameterized ten times to produce ten differently parameterized BNs for each selected topology. This yields a total of 100 test models. Fig. 10 shows two examples of BNs used for this paper.

The results averaged among 100 trials are shown in Table III, where the closeness is defined as the relative difference in mutual information between the solution from our approach and the solution from the brute-force approach. It can be seen that the solution with our method is close to the sensor selection results using the brute-force method. For further comparison, the run time of the two methods is measured on a 2.0-GHz computer, and the run time averaged among ten trials is summarized in Table III. Compared with the brute-force method, our method significantly reduces the computation time with minimum loss in sensor selection accuracy.

To demonstrate the improvement of the proposed method over random sensor selection, the results of random sensor selection are also included in Table III. For a fair comparison, we first use our method to select a best subset and then use the random method to select a subset of the same size using the same criterion. To account for the random nature of random selection, the results are averaged, and the averaged result is used to compare against the result from our method. Compared with the random sensor selection, our method shows a significant improvement in sensor selection accuracy.

Finally, we want to note that the randomly generated BN topologies (for example, the BNs in Figs. 9 and 10) may not necessarily satisfy the assumption needed for Theorem 1 to hold. Despite this, the selected sensors remain close (in mutual information) to those selected by the brute-force method, as demonstrated in Table III. We also repeat the above experiments by using naive BNs. The parameters of BNs are randomly generated. We selected k sensors from n sensors $(k < n)$ without considering sensor costs. The sensors selected by the brute-force method and by our approach have no difference.

V. Conclusion

It is computationally difficult to identify an optimal sensor subset with the information-theoretic criterion. To address problem, we have presented an approximation method to find a near-optimal sensor subset by utilizing the sensor pairwise information to infer the synergy among sensors. Specifically, this paper includes the following aspects: First, we propose to use a BN to represent sensors, their dependencies, and their relationships to other latent variables. In addition, the built-in conditional independence assumptions with the BNs allow factorizing the joint probabilities so that fusion can efficiently be performed. Second, we introduce a statistical measure to quantify the pairwise synergy among sensors. Based on the synergy measure, a synergy graph is constructed, which is used to infer synergy among multiple sensors, based on which we can then eliminate many unpromising sensor combinations. Finally, for the remaining sensor combinations, a greedy approach is introduced to identify the optimal sensor combination based on the LUB of the joint mutual information. The use of the LUB of the joint mutual information instead of the joint mutual information itself significantly reduces the computation time with minimum loss in accuracy. We demonstrate both the optimality and the efficiency of the proposed method through many random simulations under different numbers of sensors and different relationships among sensors.

A major assumption of this paper is that the two sensors are conditionally independent of each other, given another sensor between the two sensors and the fusion result. This assumption could limit the utility of this paper. As part of the future research, we will study ways to overcome this assumption. Another assumption we made in this paper is that all the sensors have the same cost. Such an assumption is not realistic for many applications. Overcoming this assumption, however, requires incorporating the sensor cost into the proposed synergy function, which is a nontrivial task. We will study this issue in the future as well.

Appendix

In the following, we introduce our proof for Theorems 1 and 2.

A. Proof of Theorem 1

Before proving Theorem 1, we give the following lemma.

Lemma 1 (Chain Rule of Mutual Information): Letting X, $Y_1, \ldots, Y_m$ be random variables, then

$$I(X; Y_1, \ldots, Y_m) = I(X; Y_1) + \sum_{i=2}^{M} I(X; Y_i \,|\, Y_1, \ldots, Y_{i-1}). \tag{13}$$

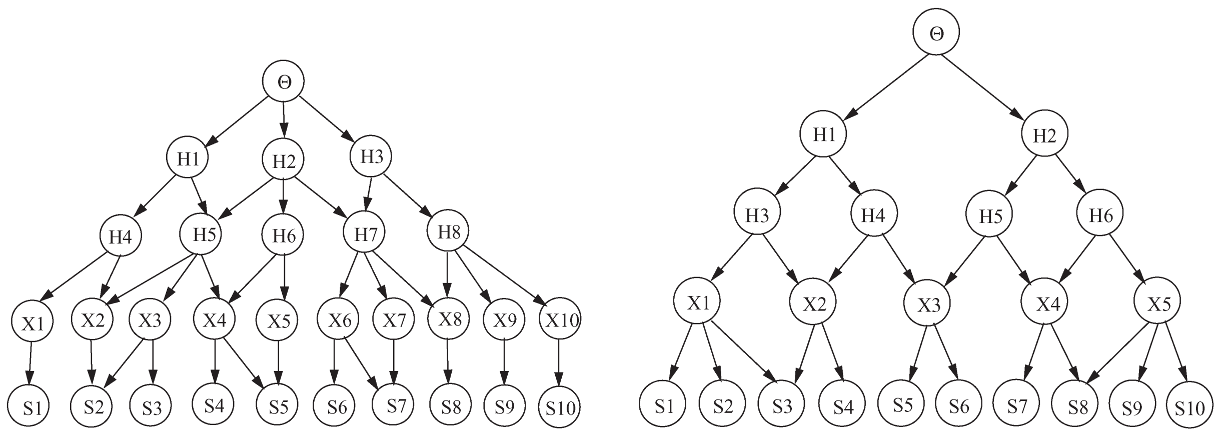


Fig. 10. Two specific examples of BN structures with different numbers of sensors used for the evaluation.

TABLE III
COMPARISON OF THE PROPOSED METHOD AND THE BRUTE-FORCE METHOD

Number	Our Approach		Random Method	Brute-Force
of	Relative mutual information difference	Run time	Relative mutual information difference	Run time
Sensors	of our method to brute force methods	(Seconds)	of random method to brute force methods	(Seconds)
7	1.56%	1.020	21.13%	63.87
8	1.77%	1.099	28.32%	355.05
9	2.75%	1.209	36.54%	2967.36
10	1.89%	1.430	39.19%	13560.54

The proof of Lemma 1 is straightforward [2]. We now turn to proving Theorem 1.

Proof: Based on Lemma 1, we have

$$\begin{aligned} I(\Theta; S_1, \ldots, S_m) &= I(\Theta; S_1) + I(\Theta; S_2 \mid S_1) + I(\Theta; S_3 \mid S_1, S_2) \\ &\quad + I(\Theta; S_4 \mid S_1, S_2, S_3) + \cdots + I(\Theta; S_m \mid S_1, \ldots, S_{m-1}). \end{aligned} \tag{14}$$

We start with an MSC containing four random variables $\{\Theta, S_1, S_2, S_3\}$, then extend it to five variables, and finally to a finite number of arbitrary random variables forming an MSC. Notice that Θ is the hypothesis, and S_1, S_2, S_3 are the sensors.

Based on Definition 6 of MSC, S_1 and S_3 are conditionally independent given S_2, i.e., $P(S_3 \mid S_1, S_2) = P(S_3 \mid S_2)$. First, we prove that the following equation holds when S_1 and S_3 are conditionally independent given S_2:

$$I(\Theta; S_3 \mid S_2) = I(\Theta; S_3 \mid S_1, S_2) \tag{15}$$

$$\begin{aligned} I(\Theta; S_3 \mid S_1, S_2) &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(\theta, s_3 \mid s_1, s_2)}{p(\theta \mid s_1, s_2) p(s_3 \mid s_1, s_2)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(\theta, s_3, s_1, s_2)}{p(\theta, s_1, s_2) p(s_3 \mid s_2)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(s_3 \mid \theta, s_1, s_2)}{p(s_3 \mid s_2)} \right\} \\ &= \sum_{\Theta, S_1, S_2, S_3} p(\theta, s_1, s_2, s_3) \left\{ \lg \frac{p(s_3 \mid \theta, s_2)}{p(s_3 \mid s_2)} \right\} \\ &= \sum_{\Theta, S_2, S_3} p(\theta, s_2, s_3) \left\{ \lg \frac{p(s_3 \mid \theta, s_2)}{p(s_3 \mid s_2)} \right\} \\ &= \sum_{\Theta, S_2, S_3} p(\theta, s_2, s_3) \left\{ \lg \frac{p(s_3, \theta \mid s_2)}{p(\theta \mid s_2) p(s_3 \mid s_2)} \right\} \\ &= I(\Theta; S_3 \mid S_2). \end{aligned} \tag{16}$$

Please note that for the derivations in (16), we assume that $p(S_3 \mid \Theta, S_1, S_2) = p(S_3 \mid \Theta, S_2)$, i.e., S_3 and S_1 are conditionally independent given both Θ and S_2, where Θ is a random variable representing the fusion result, and S_i is a sensor. The typical relationships between Θ and S_i are illustrated in Fig. 1, where Θ is typically the root node, and S_i's are the leaf nodes in the BN. Given this understanding, if the BN is such that the path (undirected path) between two sensor nodes (e.g., S_1 and S_3) goes through Θ node (e.g., the BN in Fig. 1), then following the D-separation principle for BN, $p(S_3|\Theta, S_1, S_2) = p(S_3|\Theta, S_2)$ holds. Please note that this assumption only holds for some BNs, such as the one in Fig. 1 and the naive BN. It may not hold for an arbitrary BN.

From the chain rule of mutual information, we have

$$I(\Theta; S_3, S_2) = I(\Theta; S_2) + I(\Theta; S_3 \mid S_2). \tag{17}$$

Hence, combining (16) and (17) yields

$$I(\Theta; S_3 \mid S_1, S_2) = I(\Theta; S_2, S_3) - I(\Theta; S_2). \tag{18}$$

Now, we want to apply the similar algebraic process to prove $I(\Theta; S_4 \mid S_1, S_2, S_3) = I(\Theta; S_4 \mid S_3)$ in (19), shown at the bottom of the page, given the Markov conditions that $P(S_4 \mid S_2, S_3) = P(S_4 \mid S_3)$, $P(S_1 \mid S_3, S_2) = P(S_1 \mid S_2)$, $P(S_4 \mid S_1, S_2) = P(S_4 \mid S_2)$, and $P(S_4 \mid S_1, S_3) = P(S_4 \mid S_3)$. By mutual information chain rule, we have $I(\Theta; S_3, S_4) = I(\Theta; S_3) + I(\Theta; S_4 \mid S_3)$, i.e.,

$$I(\Theta; S_4 \mid S_3) = I(\Theta; S_3, S_4) - I(\Theta; S_3). \tag{20}$$

Combining (19) and (20) produces

$$\begin{aligned} I(\Theta; S_4 \mid S_1, S_2, S_3) &= I(\Theta; S_4 \mid S_3) \\ &= I(\Theta; S_3, S_4) - I(\Theta; S_3). \end{aligned} \tag{21}$$

Finally, we can generalize the above process to prove

$$\begin{aligned} &I(\Theta, S_m \mid S_1, S_2, \ldots, S_{m-1}) \\ &\qquad = I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}). \end{aligned} \tag{22}$$

Substituting the results in (18), (21), and (22) into (14) yields

$$\begin{aligned} &I(\Theta; S_1, S_2, \ldots, S_m) \\ &\quad = I(\Theta; S_1) + I(\Theta; S_2 \mid S_1) + I(\Theta; S_3, S_2) \\ &\qquad - I(\Theta; S_2) + I(\Theta; S_3, S_4) - I(\Theta; S_3) + \cdots \\ &\qquad + I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}) \\ &\quad = I(\Theta; S_1) + I(\Theta; S_2, S_1) - I(\Theta; S_1) + I(\Theta; S_3, S_2) \\ &\qquad - I(\Theta; S_2) + I(\Theta; S_3, S_4) - I(\Theta; S_3) + \cdots \\ &\qquad + I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}) \\ &\quad = I(\Theta; S_1) + \sum_{i=2}^{M} I(\Theta; S_{m-1}, S_m) - I(\Theta; S_{m-1}). \end{aligned} \tag{23}$$

This completes the proof for Theorem 1. ■

B. *Proof of Theorem 2*

Proof: We want to prove

$$I(\Theta; S_1, \ldots, S_m) \le I^M(\Theta; S_1, \ldots, S_m). \tag{24}$$

From the mutual information chain rule, we have

$$\begin{aligned} &I(\Theta; S_1, S_2, \ldots, S_m) \\ &\quad = I(\Theta; S_1) + I(\Theta; S_2 \mid S_1) \\ &\qquad + I(\Theta; S_3 \mid S_1, S_2) + I(\Theta; S_4 \mid S_1, S_2, S_3) + \cdots \\ &\qquad + I(\Theta; S_m \mid S_1, S_2, \ldots, S_{m-1}). \end{aligned} \tag{25}$$

By Theorem 1, we have

$$\begin{aligned} &I^M(\Theta; S_1, \ldots, S_m) \\ &\quad = I(\Theta; S_1) + I(\Theta; S_2 \mid S_1) + I(\Theta; S_3 \mid S_2) \\ &\qquad + I(\Theta; S_4 \mid S_3) + \cdots + I(\Theta, S_m \mid S_{m-1}). \end{aligned} \tag{26}$$

By the definition of mutual information, we have

$$I(\Theta; S_3; S_2; S_1) = I(\Theta; S_3; S_2) - I(\Theta; S_3; S_2 \mid S_1) \tag{27}$$

which readily leads to

$$I(\Theta; S_3; S_2 \mid S_1) = I(\Theta; S_3 \mid S_2) - I(\Theta; S_3 \mid S_2, S_1). \tag{28}$$

Hence

$$I(\Theta; S_3 \mid S_2, S_1) = I(\Theta; S_3 \mid S_2) - I(\Theta; S_3; S_2 \mid S_1). \tag{29}$$

Therefore

$$I(\Theta; S_3 \mid S_2) \ge I(\Theta; S_3 \mid S_1, S_2).$$

Please note that we assume here that $I(\Theta; S_3; S_2 \mid S_1) > 0$, which is correct since for our application Θ (the hypothesis)

$$\begin{aligned} &I(\Theta; S_4 \mid S_1, S_2, S_3) \\ &\quad = \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(\theta, s_4 | s_1, s_2, s_3)}{p(\theta \mid s_1, s_2, s_3) p(s_4 | s_1, s_2, s_3)} \right\} \\ &\quad = \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(\theta, s_4, s_1, s_2, s_3)}{p(\theta, s_1, s_2, s_3) p(s_4 | s_3)} \right\} \\ &\quad = \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_1, s_2, s_3)}{p(s_4 | s_3)} \right\} \\ &\quad = \sum_{\Theta, S_1, S_2, S_3, S_4} p(\theta, s_1, s_2, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_3)}{p(s_4 | s_3)} \right\} \\ &\quad = \sum_{\Theta, S_3, S_4} p(\theta, s_3, s_4) \left\{ \lg \frac{p(s_4 | \theta, s_3)}{p(s_4 | s_3)} \right\} \\ &\quad = \sum_{\Theta, S_2, S_3} p(\theta, s_3, s_4) \left\{ \lg \frac{p(s_4, \theta \mid s_3)}{p(\theta \mid s_3) p(s_4 | s_3)} \right\} = I(S_4; \Theta \mid S_3) = I(\Theta; S_4 \mid S_3) \end{aligned} \tag{19}$$

and the other variables (sensors) are not independent of each other.

Similarly, we have $I(\Theta; S_4 \mid S_3) \geq I(\Theta; S_4 \mid S_1, S_2, S_3)$ and $I(\Theta; S_m \mid S_{m-1}) \geq I(S_m \mid S_1, S_2, \ldots, S_{m-1})$.

Hence, (26) $\geq$ (25). The equality sign holds when the Markov property between neighbor sensors is true.

Hence, this completes the proof for Theorem 2. ■

Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable and constructive comments, which help to significantly improve this paper.

References

[1] Y. Zhang and Q. Ji, "Active and dynamic information fusion for multisensor systems with dynamic Bayesian networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 2, pp. 467–472, Apr. 2006.

[2] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[3] D. E. Hecherman, E. Horvitz, and B. N. Nathwani, "Toward normative expert systems: Part I. The pathfinder project," *Methods Inf. Med.*, vol. 31, no. 2, pp. 90–105, Jun. 1992.

[4] L. van der Gaag and M. Wessels, "Selective evidence gathering for diagnostic belief networks," *AISB Q.*, vol. 86, pp. 23–34, 1993.

[5] S. Dittmer and F. Jenson, "Myopic value of information in influence diagrams," in *Uncertainty Artif. Intell.*, 1997, pp. 142–149.

[6] V. Bayer-Zubek, "Learning diagnostic policies from examples by systematic search," in *Uncertainty Artif. Intell.*, 2004, pp. 27–34.

[7] V. Krishnamurthy, "Algorithms for optimal scheduling and management of hidden Markov model sensors," *IEEE Trans. Signal Process.*, vol. 50, no. 6, pp. 1382–1397, Jun. 2002.

[8] E. Ertin, J. W. Fisher, and L. C. Potter, "Maximum mutual information principle for dynamic sensor query problems," in *Proc. Inf. Process. Sens. Netw.*, San Francisco, CA, 2003, pp. 405–416.

[9] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proc. Inf. Process. Sens. Netw.*, Berkeley, CA, 2004, pp. 36–45.

[10] D. Heckerman, E. Horvitz, and B. Middleton, "An approximate nonmyopic computation for value of information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 3, pp. 292–298, Mar. 1993.

[11] A. Krause and C. Guestrin, "Optimal nonmyopic value of information in graphical models—Efficient algorithms and theoretical limits," in *Int. Joint Conf. Artif. Intell.*, 2005, pp. 1339–1345.

[12] A. Krause and C. Guestrin, "Near-optimal nonmyopic value of information in graphical models," in *Uncertainty Artif. Intell.*, 2005, pp. 324–333.

[13] W. Liao and Q. Ji, "Efficient active fusion for decision-making via voi approximation," in *21th Nat. Conf. Artif. Intell.*, 2006, pp. 1180–1185.

[14] Y. Zhang and Q. Ji, "Sensor selection for active information fusion," in *20th Nat. Conf. Artif. Intell.*, 2005, pp. 1229–1234.

[15] W. Liao, W. Zhang, and Q. Ji, "A factor tree inference algorithm for Bayesian networks and its application," in *Proc. 16th ICTAI*, Boca Raton, FL, 2004, pp. 652–656.

[16] W. Zhang and Q. Ji, "A factorization approach to evaluating simultaneous influence diagrams," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 36, no. 4, pp. 746–757, Jul. 2006.

[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. San Mateo, CA: Morgan Kaufmann, 1988.

[18] F. V. Jensen and F. Jensen, "Optimal junction trees," in *Proc. 10th Conf. Uncertainty AI*, San Francisco, CA, 1994, pp. 360–366.

[19] R. Dechter, "Bucket elimination: A unifying framework for probabilistic inference," in *Proc. 12th Conf. Uncertainty AI*, San Francisco, CA, 1996, pp. 211–219.

Yongmian Zhang (M'04) received the Ph.D. degree in computer engineering from the University of Nevada, Reno, in 2004.

He holds a research position with the Department of Electrical, Computer and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. His areas of research include information fusion, computer vision, and human–computer interactions.

Qiang Ji (SM'04) received the Ph.D. degree in electrical engineering from the University of Washington, Seattle.

He is currently a Professor with the Department of Electrical, Computer, and Systems Engineering and the Director of the Intelligent Systems Laboratory, Rensselaer Polytechnic Institute, Troy, NY. He is also a Program Director with the National Science Foundation (NSF), Arlington, VA, where he is responsible for managing part of the NSF's computer vision and machine learning programs. He previously held teaching and research positions with Beckman Institute, University of Illinois, Urbana; Robotics Institute, Carnegie Mellon University, Pittsburgh, PA; the Department of Computer Science, University of Nevada, Reno; and the U.S. Air Force Research Laboratory. His research has been supported by major governmental agencies, including the NSF, the National Institutes of Health, the Defense Advanced Research Projects Agency, the Office of Naval Research, the Army Research Office, and the Air Force Office of Scientific Research, as well as by major companies, including Honda and Boeing. He has published more than 150 papers in peer-reviewed journals and conference proceedings. He is an Editor for several computer-vision- and pattern-recognition-related journals.

Prof. Ji has served as program committee member, area chair, and program chair in numerous international conferences/workshops.

Non-myopic Active Learning with Mutual Information

Yue Zhao

Department of Automation

Minzu University of China
Beijing,100081, China

zhaoyueso@muc.edu.cn

Qiang Ji

Department of Electronical,Computer,and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180, USA

qji@ecse.rpi.edu

***Abstract* - Active learning methods seek to reduce the number of labeled instances needed to train an effective classifier. Most current methods are myopic, i.e. select a single unlabelled sample to label at a time. The batch-mode active learning methods, on the other hand, typically select top N unlabeled samples with maximum score. Such selected samples often cannot guarantee the learner's performance. In this paper, a non-myopic active learning algorithm is presented based on mutual information. Our algorithm selects a set of samples at each iteration, and the objective function of the algorithm is proved to be submodular, which guarantees to find the near-optimal solution. Our experimental results on UCI data sets show that the proposed algorithm outperforms myopic active learning.**

Index Terms - Non-myopic active learning, Mutual information, Submodular function.

I. Introduction

Most of the current study in active learning has focused on selecting a single unlabeled sample in each iteration. In this case, the classifier has to be retained after each selected sample is labeled. These myopic approaches are also not suited to the parallel labeling environment. By contrast, batch-mode active learning methods select a set of unlabeled samples at a time. They provide a solution to multiple annotators and model with slow training procedures. A simple strategy toward selecting a query batch is to myopically query the top N queries according to a given query strategy. Methods based on such a strategy do not work well, since they greedily select the "best" unlabeled samples at each iteration in local and fail to consider the overlap in information content among the "best" instances [1].

To address this issue, a few improved batch-mode active learning algorithms have been proposed. Brinker [2] considers an approach for SVMs that explicitly incorporates diversity among instances in the batch. Xu et al. [3] propose a similar approach for SVM active learning, which also incorporates a density measure. Specifically, they query cluster centroids for instances that lie close to the decision boundary. Hoi et al. [4] extend the Fisher information framework to the batch-mode setting for binary logistic regression. But most of these approaches still use greedy heuristics to ensure that instances in the batch are both diverse and informative without a guarantee of near-optimal solution.

In this paper, we present a non-myopic active learning algorithm to query instances in groups. In active learning one solution to decide which sample to label is based on its effect on the remaining unlabeled data. So, our aim is to select a set of unlabeled samples which, if labeled, can maximize the confidence (certainty) on remaining unlabeled data. Based on this idea, we exploit mutual information to construct the objective function.

The objective function of our algorithm is proved to be submodular, which guarantees the greedy method can find near-optimal sets to be added into training data so that active learner can improve in both accuracy and efficiency.

This paper is structured as follows. Section II gives the description of the objective function of our non-myopic active learning method and the proof of submodularity of the objective function. Section III presents our algorithm. Section IV contains experimental results on Chess database and Tic-Toe-Tac Endgame database from the UCI Machine Learning Repository. Section V summarizes the contributions made in this paper.

II. The objective function and its submodularity

The aim of our non-myopic active learning is to select a subset of K unlabeled samples in each iteration, such that when the samples are given true labels and added to the training set D, the learner trained on the augmented training set can result in the maximum classification uncertainty reduction on the test set. So the objective function is as follows,

$$A = \arg\max_{A\subset U} \hat{R}(A), s.t. |A| \le K, \tag{1}$$

where A is a subset of unlabelled training data, and U is the unlabelled data pool. For a set A, the uncertainty reduction function $\hat{R}(A)$, i.e., the mutual information *criterion* between A and U can be defined as

$$\hat{R}(A) = \hat{H}_D(U) - \hat{H}_D(U \setminus A | A), \tag{2}$$

where $\hat{H}_D(.)$ is the entropy for measuring the classification uncertainty on the remaining unlabelled data given a current learner. Assuming the initial small set of labelled data D, a large pool of unlabeled data $U = \{x_1, x_2, \ldots, x_M\}$, and $x_l \in U$, $\hat{H}_D(.)$ can be computed by

$$\hat{H}_D(U) = \frac{1}{|U|} \sum_{x_l \in U} \sum_{y_l \in Y} -(\hat{P}_D(y_l | x_l) \log \hat{P}_D(y_l | x_l)). \tag{3}$$

Let $A=\{x_1,\ldots,x_m\}$ be a subset of U and $x_k \in U \setminus A$, $\hat{H}_D(U \setminus A \mid A)$ is measuring the expected uncertainty on the remaining data set $U \setminus A$ after A is selected. Since before we make the query for set A, $y_1,\cdots,y_m$, the true labels for each $x_1,\cdots,x_m$, are unknown, so the conditional entropy $\hat{H}_D(U \setminus A \mid A)$ can be written as

$$\hat{H}_D(U \setminus A \mid A)=\frac{1}{|U-A|}\frac{1}{|Y|^m}\sum_{x_k\in U\setminus A}\sum_{y_k\in Y}\sum_{y_1\in Y}\cdots\sum_{y_m\in Y}-(\hat{P}_D(y_k \mid x_1,y_1,\ldots,x_m,y_m,x_k)\log\hat{P}_D(y_k \mid x_1,y_1,\ldots,x_m,y_m,x_k)). \tag{4}$$

In Equation 4, $\hat{P}_D(y_k \mid x_1,y_1,\ldots,x_m,y_m,x_k)$ is the estimated output distribution given D, x_k and some possible labels $y_1,\cdots,y_m$ for $x_1,\cdots,x_m$, i.e.,

$$\hat{P}_D(y_k \mid x_1,y_1,\ldots,x_m,y_m,x_k)=\hat{P}_{D+\{x_1,y_1,\ldots,x_m,y_m\}}(y_k \mid x_k). \tag{5}$$

In [5], it showed that, if the objective function F is submodular, then a greedy algorithm, which starts with the empty set $A=\Phi$, and iteratively adds to A the element $s^*=\arg\max_s F(A)$ until k elements have been selected, finds a near-optimal set. Its result uses the concept of submodularity, i.e., an intuitive diminishing returns property: a new observation decreases our uncertainty more if we know less. According to this conclusion, we give a proof of the submodularity of $\hat{R}$ as follows.

Proof of the submodularity of $\hat{R}$ ***:***

Let $A \subset B \subseteq U$ and unselected samples $X \notin A$, and if $\hat{R}$ is submodular, then it should hold that "diminishing returns" property,

$$\hat{R}(A\cup X)-\hat{R}(A)\geq\hat{R}(B\cup X)-\hat{R}(B). \tag{6}$$

Since $\hat{H}_D(.)$ is the entropy of the learner's posterior, it holds the property of entropy, i.e., the 'information never hurts' principle [6],

$$\hat{H}_D(X \mid A)\leq\hat{H}_D(X), \tag{7}$$

i.e., adding A with true labels cannot increase the entropy [7]. Because the marginal increase of $\hat{R}$ can be written as

$$\begin{aligned}
&\hat{R}(A\cup X)-\hat{R}(A)\\
&=\hat{H}_D(U)-\hat{H}_D(U \setminus A\cup X \mid A\cup X)\\
&\quad-\hat{H}_D(U)+\hat{H}_D(U \setminus A \mid A)\\
&=\hat{H}_D(U \setminus A \mid A)-\hat{H}_D(U \setminus A\cup X \mid A\cup X)\\
&=\hat{H}_D(U)-\hat{H}_D(A)-\hat{H}_D(U)+\hat{H}_D(A\cup X)\\
&=\hat{H}_D(A\cup X)-\hat{H}_D(A)\\
&=\hat{H}_D(X \mid A).
\end{aligned} \tag{8}$$

In Equation 8, we used the chain rule of the joint entropy, i.e.

$$H(Y \mid X)=H(X,Y)-H(X). \tag{9}$$

Submodularity is simply a consequence of information never hurts principle:

$$\hat{R}(A\cup X)-\hat{R}(A)=\hat{H}_D(X \mid A)\geq\hat{H}_D(X \mid B)=\hat{R}(B\cup X)-\hat{R}(B). \tag{10}$$

So we prove that $\hat{R}$ is submodular.

III. A NON-MYOPIC ACTIVE LEARNING ALGORITHM

The problem of maximizing submodular function is NP-hard. The greedy algorithm is an efficient algorithm to reduce the computation, and achieves near-optimal results if submodularity property holds. We exploit the greedy algorithm to iteratively find each sample for a batch A. In the step of greedily finding A, selecting a sample will be stopped when the current $\hat{H}_D(U \setminus A \mid A)$ is larger than the previous one or when the number of selected samples is larger than K.

After selecting a set A, the algorithm adds A with true labels into the labeled training data. The learner is retrained on new training set. If the prediction accuracy of current learner is not satisfied, the process is repeated.

Algorithm:

Initialization: Randomly select a small set D from unlabeled samples pool, then assign a class label to each of them, next construct an initial training set. Train the classifier C using D.

While stopping criterion (here prediction accuracy) is not satisfied

1. ***Greedily find A;***

2. Add A with true labels to D to form D';

3. Retrain the classifier C from D', and obtain its prediction accuracy on test date set.

Because the $\hat{H}_D(U)$ is always the constant while iteratively adding x_m to A, we can get the unselected sample x_m which makes the minimum $\hat{H}_D(U \setminus A \mid A)$ in the process of finding A. The process of ***Greedily find A*** is as follows.

Greedily find A:

```
$A = \{\}$;
While $|A| \le K$
Do
    $x_m^* = \arg\min_{x_m \in U \setminus A} \hat{H}_D(U \setminus A | A)$;
If $\hat{H}_D(U \setminus A \cup \{x_m^*\} | A \cup \{x_m^*\}) < \hat{H}_D(U \setminus A | A)$
    $A = A + \{x_m^*\}$;
Else
    Break;
End
```

IV. Experimental Results

Two benchmark data sets from UCI Machine Learning Repository are used to evaluate the performance of our algorithm. They are for binary classification task. We choose the TAN classifier as a classification algorithm. In the step of ***Greedily find A***, let $K = 3$. To evaluate the performance of our approach, we compare the results of our approach (non-myopic) with the results from the expected log loss reduction algorithm of myopic active learning and "N-best" batch method.

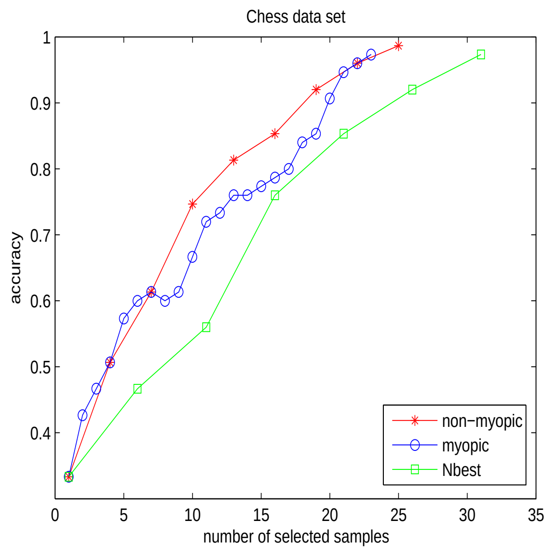


Fig. 1 The comparison results by running Chess data set.

The first data set is from chess (King-Rook vs King-Pawn) database. The data set is randomly partitioned into training set of 75 instances including 22 initially labeled examples and 53 unlabeled examples. The independent test set consists of 85 instances.

Figure 1 shows the resulting accuracy of three algorithms as the function of number of selected samples. The maximum possible accuracy is 98% after all the unlabeled data has been labeled. It can be seen that after a few queries (6 queries) our algorithm (non-myopic) can have a higher accuracy than myopic active learning and N-best samples methods. And after 8 iterations our non-myopic active learning gets 98% accuracy with the classifier is retrained eight times. However myopic active learning needs to retrain the classifier 23 times for 97% accuracy. Our non-myopic active learning is more efficient and accurate than myopic active learning and N-best methods.

The second data set for our experiment is from Tic- Toe-Tac database. The data set is randomly partitioned into training set of 203 instances, in which 59 labeled examples and 144 unlabeled examples are included, and independent test set of 172 instances.

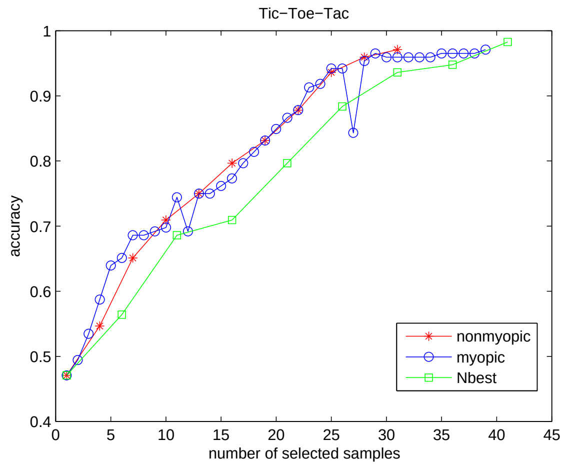


Fig. 2 The comparison results by running tic-toe-tac data set.

The comparison results are given in Figure 2. The maximum possible accuracy is 97% after all the queries. After 10 iterations, our algorithm reaches 97% accuracy. However myopic active learning needs to retrain the classifier 39 times for 97% accuracy. Again, the results show that non-myopic active learning approach outperforms myopic active learning approach and N-best method.

V. Conclusion

Most current methods for active learning myopically select the 'best' sample or 'N-best' samples in local to label. These methods cannot achieve the near-optimal results, and does not work well. By contrast, non-myopic active learning can query samples in groups at each iteration. If the objective function is submodular, then it can find near-optimal sets and has a better performance than myopic active learning. Meanwhile it is efficient in training model and suited to parallel labeling environment. In this paper we proposed a non-myopic active learning based on the mutual information, and proved the objective function is submodular. So the experimental results show that it outperforms myopic active learning.

References

[1] B. Settles, "Active Learning Literature Survey," *Computer Sciences Technical Report 1648*, University of Wisconsin-Madison, January 9, 2009.

[2] K. Brinker, "Incorporating diversity in active learning with support vector machines," *In Proceedings of the International Conference on Machine Learning (ICML)*, pages 59–66. AAAI Press, 2003.

[3] Z. Xu, R. Akella, and Y. Zhang, "Incorporating diversity and density in active learning for relevance feedback," *In Proceedings of the European Conference on IR Research (ECIR)*, pages 246–257. Springer-Verlag, 2007.

[4] S.C.H. Hoi, R. Jin, and M.R. Lyu, "Large-scale text categorization by batch mode active learning," *In Proceedings of the International Conference on the World Wide Web*, pages 633–642. ACM Press, 2006a.

[5] C. Guestrin, A. Krause and A. Singh, "Near-optimal sensor placements in Gaussian processes," *ICML*, pages 265 – 272, 2005.

[6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Interscience, 1991.

[7] C. Guestrin, A. Krause, and A. Singh, "Near-optimal sensor placements in gaussian processes," *ICML*, 2005.

An Active Learning Method under Very Limited Initial Labeled Data

Yue Zhao

Department of Automation

Minzu University of China
Beijing,100081, China

zhaoyueso@muc.edu.cn

Qiang Ji

Department of Electronical,Computer,and System Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180, USA

qji@ecse.rpi.edu

Abstract **- Active learning methods seek to reduce the number of labeled instances needed to train an effective classifier. Most current methods assume the availability of some reasonable amount of initially labeled training data so that the learners can be trained with sufficient quality. However, for many applications, the amount of initial training data is often limited, this will affect the quality of the initial learners, which, in turn, affect the performance of the active learning methods. In this paper, we introduce a new non-parametric active learning strategy that can perform well even under very limited initial training data. Our method selects the query instance that simultaneously maximizes its label uncertainty and the classification accuracy on the unlabelled test data. Our method hence avoids selecting outliers and does not require good initial learner. The experimental results with benchmark datasets show that our method outperforms state of the art methods especially when the amount of the initially labeled data is small or when the quality of the initially labeled data is poor.**

Index Terms - Active learning. Minimal total entropy reduction. Limited initial labeled data.

I. INTRODUCTION

Active learning aims to achieve greater accuracy with fewer labeled training instances by selecting the most informative data to label [1]. Active learning selects unlabeled examples for labeling if the predicted label is highly uncertain. Based on this view, some existing works in active learning have concentrated on two approaches: Uncertainty-based method [3], [4], [5] and committee-based method [6]. The former estimates sample uncertainty using one classifier while the latter does so by a committee of classifiers. Both approaches examine each unlabeled sample one at a time, often independent of the remaining unlabeled samples. Although both approaches can get to the classification boundary fast, they are also susceptible to outliers since outliers are also often uncertain. In fact, studies show that both uncertainty sampling and the committee-based methods often fail by selecting outliers [7], [8].

To alleviate this problem, one solution to decide which sample to label is not only based on the properties of selected sample but also on its effect on the remaining unlabeled data. So, from a different perspective, if the instance to be labeled can maximize the confidence (certainty), i.e. makes the current learner to have a good generalization error over the unlabeled data, the instance with true label to be added into labeled data set can improve the performance of the classifier. Based on this idea, Roy and McCallum first proposed the estimated log loss reduction (ELLR) framework for text classification using naive Bayes [7]. Zhu et al. combined this framework with a semi-supervised learning approach, resulting in a dramatic improvement over random or uncertainty sampling [8]. Guo and Greiner employ an “optimistic” variant [9], but their formulation is, in fact, equivalent to minimizing the expected future log loss. The ELLR framework has the dual advantage of being near optimal and not dependent on the model class. While effectively addressing the outlier issue, these methods suffer one major limitation. They typically require a reasonably good initial learner to start with since their methods use the $P(y \mid x_m, D)$ determined by labeled data to weight the log loss. This makes the performance of the estimated error reduction further depended on the quality of initial model [10]. These methods perform poorly for cases that have a very small amount of initial labeled training data.

To overcome the limitations of the above methods, we propose to combine the uncertainty based method with the generalization error method. The selected sample by our method considers both its own uncertainty and its potential to reduce the classification uncertainties on unlabelled data. In our method, the instance to be labeled possesses high uncertainty on its label and at the same time should maximize the classification confidence (certainty) on unlabelled data. The uncertainty of each sample can be characterized by its entropy and its effect on the reduction of the uncertainty of the unlabeled data can be characterized by its Minimal Total Entropy Reduction (MTER).

The main benefit of our algorithm, compared with the ELLR used by Roy et al [7], is that it decreases the influence of using the current learner to approximate the label distribution for weighting the log loss, which is very sensitive to the number of labeled training samples and their quality. In addition, the minimum total entropy criterion is more conservative and strict than the log loss, therefore leading to improved performance. In paper [2] a similar active learning method, Mm+M, is proposed, and it employs ’most uncertainty’ approach to change the selection rule based on minimum total entropy when it encounters an unexpected label. In fact, this method still selects a sample by using either generalization error method or uncertainty-base method but not using both criteria at the same time. Experiments show that our active learning can quickly achieve the considerable accuracy with fewer labeled samples than the state-of-the-art methods, such as ELLR, uncertainty-based, Mm+M, in particular when the quality of the initial labeled data is poor or the number of the initial training data is small. In the sections to follow, we describe our method in pool-based active learning, and its performance.

II. Active Learning with Pool-Based Sample

Pool-based active learning is an interactive learning technique designed to reduce the labour cost of labeling in which the learning algorithm can freely assign the unlabeled data to the training set. Active learning starts from an initial labeled examples and lets the learner iteratively update its training set while learning at each step from the new knowledge gain provided by newly labeled examples. An overview of pool based active learning can be seen in Figure 1. The classifier $C(\cdot)$ is trained by the labeled examples L, and then a selection function $S_f(\cdot)$ selects the most appropriate examples S from an unlabeled data pool U given the knowledge already acquired by the learner.

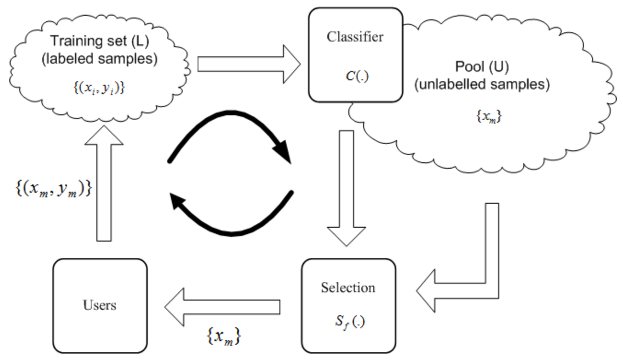


Fig. 1 An overview of pool-based active learning.

III. A New Query Strategy

Our active learning approach is based on actively identifying and annotating the samples which will result in maximum uncertainty on its label and in the meantime yields the maximum certainty on class labels of the remaining unlabeled data. Pool-based active learning typically includes a small set of labeled data $L = \{(x_1, y_1), \cdots, (x_k, y_k)\}$ and a large pool of unlabeled data $U = \{x_{k+1}, \cdots, x_m, \cdots, x_N\}$. Assume we select x_m from U for the next instance of human labelling. Our objective function is

$$x_m^* = \arg\max_{x_m \in U} M(x_m), \quad (1)$$

where $M(x_m)$ is the sum of two terms: the generalization error reduction on the unlabeled data and the label uncertainty of the selected sample. It is defined as

$$M(x_m) = R(x_m) + H(Y_m \mid x_m, L). \quad (2)$$

In Equation 2, $R(x_m)$ is to measure the error reduction of a candidate sample xm over remaining unlabeled samples, and it can be defined as

$$R(x_m) = E(Y_U \mid X_U, L) - E(Y_U \mid X_U, x_m, y_m, L). \quad (3)$$

$E(.)$ is a generalization error function. Since the first term in Equation 3 does not depend on the instance x_m selected, so we can rewrite Equation 3 as

$$R(x_m) = -E(Y_U \mid X_U, x_m, y_m, L), \quad (4)$$

and Equation 2 as

$$M(x_m) = -E(Y_U \mid X_U, x_m, y_m, L) + H(Y_m \mid x_m, L). \quad (5)$$

First, for evaluating the classification performance on the unlabeled data, i.e. generalization error, the learner is initially trained on the labeled data L. Once trained, given an input x_n from U, it produces a probability distribution on its label y_n, based on which y_n can be determined. Given x_m, its label y_m and the existing labeled data, we can construct a classifier that can estimate the probability distribution of label y_n for each unlabeled data x_n, i.e.,

$$P_{n,m} = P(y_n \mid (x_1, y_1), (x_2, y_2), \cdots, (x_k, y_k), x_m, y_m, x_n), \quad (6)$$

where $k < m, n \le N$ and $n \ne m$. The error (uncertainty) of estimated label y_n for input x_n can be characterized by its entropy

$$H_{n,m} = -\sum_{y_n \in Y} P_{n,m} \log P_{n,m}, \quad (7)$$

where Y is all possible outcome labels. Then, the total entropy for all remaining unlabeled data in the pool given (x_m, y_m) can be computed as

$$E(Y_U \mid X_U, x_m, y_m, L) = \sum_{\substack{n=k+1 \\ n \ne m}}^{N} H_{n,m}. \quad (8)$$

Thus one criterion of our active learning approach is to select a query, x_m , such that when the query is given true label y_m and added to the training set, the learner trained on the resulting set $(L+(x_m, y_m))$ results in the maximum reduction on the uncertainty of the labels of the remaining unlabeled samples in the pool, i.e. the smaller entropy for all remaining unlabeled data. Meanwhile, before we make the query, y_m , the true label for x_m , is unknown. Thus, Equation 8 can be approximated by computing the minimum estimated uncertainty for x_m over each possible label $\hat{y}_m$, i.e.

$$\hat{E}(Y_U \mid X_U, x_m, \hat{y}_m, L) = \min_{\hat{y}_m \in Y} \sum_{\substack{n=k+1 \\ n \neq m}}^{N} (\sum_{y_n \in Y} (-\hat{P}_{n,m} \log \hat{P}_{n,m})). \quad (9)$$

x_m is the next sample to be selected for labeling, which is added into L . The learner is then retrained on L , and the process repeats until the stopping criterion is satisfied. Equation 9 is the minimum total entropy, which is different from the expected entropy used in [7]. Computing the expected entropy requires the current learner to estimate the current classifier's posterior $\hat{P}(y_m \mid x_m, L)$ for a candidate to compute the weight for each label. In the case of small number of initial labeled samples, $\hat{P}(y_m \mid x_m, L)$ cannot be estimated accurately and the expected log loss reduction computed in this case will not be accurate either. However, $\hat{P}(y_m \mid x_m, L)$ is not needed in evaluating the generalization error, our query strategy hence is less dependent on both the quality and quantity of initial labeled data set.

On the other hand, the selected sample should possess high uncertainty on its own label. Hence, the second criterion in Equation 5 can be computed by the entropy base on current learner, i.e.,

$$H(Y_m \mid x_m, L) = -\sum_{y_m \in Y} (\hat{P}(y_m \mid x_m, L) \log \hat{P}(y_m \mid x_m, L)). \quad (10)$$

Our objective function considers both the uncertainty of a candidate and its potential to reduce the uncertainties of the unlabelled data. So it selects the unlabeled sample with the maximum uncertainty and maximum classification error reduction on the unlabelled data. This therefore leads to improved performance under very limited initial labeled data as will be demonstrated in our experiments. The algorithm is summarized in Table I.

The above algorithm is computationally intensive. Several methods can be used to improve the algorithm efficiency including sampling and clustering the unlabelled data pool so that only seeds are considered for labeling. Another alternative to speedup the algorithm is to use an incremental learning mechanism to learn the classifier as the labeled data is gradually made. As our focus is on the learning method, we will not discuss this issue in this paper.

IV. Experimental Results

Two benchmark data sets from UCI Machine Learning Repository are used to evaluate the performance of our method. One is for binary classification task and the other is for multiple classification task. We choose the TAN classifier as a classification algorithm. To evaluate the performance of our approach, we compare the results of our approach (UNMTER) with the results from the expected log loss reduction algorithm (ELLR) in [7], Mm+M in [2] and most uncertainty method.

TABLE I
UN-MTER Algorithm

```
1: Initialization: Randomly select a small set of samples from unlabeled
   sample pool $U$ , assign a class to each sample of them, then construct an
   initial training set $L$ . Train the classifier $C$ using $L$ .
2: While stopping criterion (here prediction accuracy) is not satisfied
3:      Compute $\hat{E}(Y_U \mid X_U, x_m, \hat{y}_m, L)$ for each $x_m$ from $U$ using
            Equation 9;
4:      Compute $H(Y_m \mid x_m, L)$ for each $x_m$ from $U$ using Equation 10;
5:      Compute $M(x_m)$ for each $x_m$ from $U$ using Equation 5;
6:      Select $x_m^*$ with the maximum $M(x_m)$ ;
7:      Add $x_m^*$ with true label $y_m^*$ to $L$ to form $L_m^*$ , where
        $L_+^* = L + (x_m^*, y_m^*)$ ;
8:      Retrain classifier $C$ from $L_+^*$ , and obtain predication accuracy.
```

The first data set is from Tic-Tac-Toe Endgame database, which consists of 958 instances. The data set is randomly partitioned into training set of 203 instances including 10 initially labeled examples and 193 unlabeled examples. The independent test set consists of 172 instances.

Figure 2 shows the resulting accuracy of four algorithms as the function of number of selected samples. The maximum possible accuracy is 97% after all the unlabeled data has been labeled. In this experiment, each of active learners sequentially selects 45 instances from unlabeled pool and adds to the labeled set. It can be seen that after 21 queries our algorithm (UN-MTER) and Mm+M reach 65%. In contrast, the ELLR reaches 54%. After 23 queries our algorithm keeps getting the higher accuracy than Mm+M and ELLR. Meanwhile it is showed that the most uncertainty method (UN) has the worst performance. The result demonstrates that, our approach outperforms Mm+M, ELLR and the most uncertainty under very limited initial labeled data set. However, ELLR can match UN-MTER as the labeled data set contains a substantial number of samples. Nevertheless, our method is practically useful for some application domains in which the availability of initially labeled data is restricted.

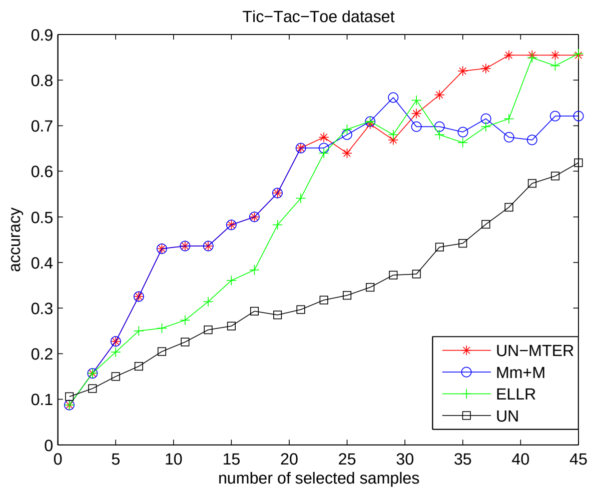


Fig. 2 The comparison results by running Tic-Tac-Toe data set.

The second data set for our experiment is from Nursery database. The data set is randomly partitioned into training set of 182 instances, in which 16 labeled examples and 166 unlabeled examples are included, and independent test set of 282 instances. This data set is for a multiple classification task with 5 class attributes.

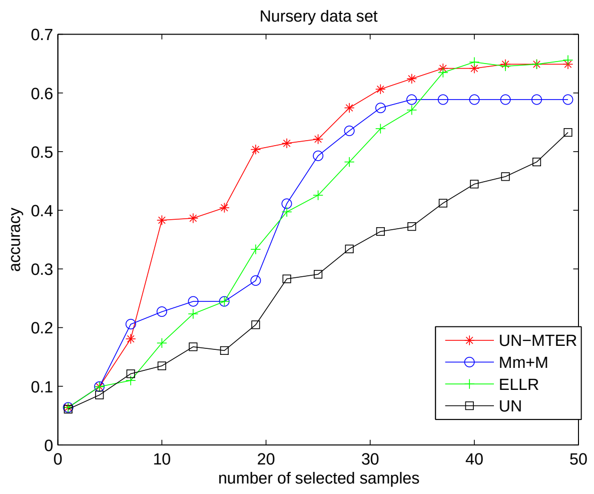


Fig. 3 The comparison results by running Nursery data set.

The comparison results are given in Figure 3. The maximum possible accuracy is 66% after all the queries. After 20 queries, our algorithm reaches 51%. ELLR and Mm+M, on the other hand, only reached 35% and 33% respectively. Since then, UN-MTER has faster speedup than other methods. Again, the results show that our approach outperforms ELLR, Mm+M and UN when the number of initial labeled data is very limited.

V. Conclusion

Most current methods for active learning assume the availability of some reasonable amount of initially labeled training data with sufficient quality. However, in many applications, the amount and the quality of initial training data are often limited. This will affect the quality of the initial learners, which, in turn, affect the performance of the active learning methods. To address this issue, we introduce the method based on maximizing the minimum total entropy reduction on the unlabeled data and maximum uncertainty of a sample on the current learner. The experimental results with benchmark datasets show that our method outperforms the state-of-the-art methods especially when the amount of the initially labeled data is small or when the quality of the initially labeled data is poor.

References

[1] B. Settles, “Active Learning Literature Survey,” *Computer Sciences Technical Report 1648*, University of Wisconsin-Madison, January 9, 2009.

[2] W.M.Hu, W. Hu, N.H. Xie, and S. Maybank, “Unsupervised Active Learning Based on Hierarchical Graph-Theoretic Clustering,” *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART B: CYBERNETICS*, VOL. 39, NO. 5, OCTOBER 2009.

[3] T. Scheffer, C. Decomain, and S. Wrobel, “Active hidden Markov models for information extraction,” *In Proceedings of the International Conference on Advances in Intelligent Data Analysis (CAIDA)*, pages 309-318, Springer-Verlag, 2001.

[4] A. Culotta and A. McCallum, “Reducing labeling effort for structured prediction tasks,” *In Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 746-751, AAAI Press, 2005.

[5] S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee, “MMR-based active machine learning for bio named entity recognition,” *In Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT- AACL)*, pages 69-72, ACL Press, 2006.

[6] I. Dagan and S. Engelson, “Committee-based sampling for training probabilistic classifiers,” *In Proceedings of the International Conference on Machine Learning (ICML)*, pages 150-157, Morgan Kaufmann, 1995.

[7] N. Roy and A. McCallum, “Toward optimal active learning through sampling estimation of error reduction,” *In Proceedings of the International Conference on Machine Learning (ICML)*, pages 441-448, Morgan Kaufmann, 2001.

[8] X. Zhu, J. Lafferty, and Z. Ghahramani, “Combining active learning and semi-supervised learning using gaussian fields and harmonic functions,” *In Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58-65, 2003.

[9] Y. Guo and R. Greiner, “Optimistic active learning using mutual information,” *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 823-829, AAAI Press, 2007.

[10] A. I. Schein and L.H. Ungar, “A-Optimality for Active Learning of Logistic Regression Classifiers,” *The University of Pennsylvania Department of Computer and Information Science Technical Report*, No. MS-CIS-04-07.

Improving Bayesian Network Parameter Learning using Constraints

Cassio P. de Campos
Rensselaer Polytechnic Institute
decamc@rpi.edu

Qiang Ji
Rensselaer Polytechnic Institute
jiq@rpi.edu

Abstract

This paper describes a new approach to unify constraints on parameters with training data to perform parameter estimation in Bayesian networks of known structure. The method is general in the sense that any convex constraint is allowed, which includes many proposals in the literature. Driven by a maximum entropy criterion and the Imprecise Dirichlet Model, we present a constrained convex optimization formulation to combine priors, constraints and data. Experiments indicate benefits of this framework.

1 Introduction

Bayesian Networks (BNs) encode a joint probability distribution for a set of random variables in a compact graph structure. The problem of parameter learning concerns the estimation of probability measures of conditional probability distributions, given the graph structure of the BN. Many techniques depend heavily on training data. Ideally, with enough data, it is possible to learn parameters by standard statistical analysis like maximum likelihood (ML) estimation. However, data may be insufficient, leading to inaccurate estimations.

This paper proposes a framework for the parameter learning problem that combines data and domain knowledge in the form of constraints. There are two types of constraints: *soft constraints* on priors and *hard constraints* on estimations. Driven by the Imprecise Dirichlet Model [17], prior beliefs are encoded using a set of Dirichlet distributions. Combined with data and constraints on estimations, the result is a set of estimations, on which we apply the maximum entropy principle to obtain a single estimation. Constraints on estimations are viewed as *hard* constraints and assumed to be correct. Thus only general and certainly valid constraints shall be used. On the other hand, constraints on priors are *soft* because estimations are adapted and corrected by training data even if some *soft* constraints are wrongly stated. Altogether, we can encode constraints that are certain as well as less reliable constraints.

There are many approaches to parameter learning of BNs using constraints. For instance, penalty functions can be employed [1], but global optimality is not always guaranteed. Isotonic regression is also applicable, but its complexity is high [7]. Non-convex optimization also leads to high complexities [5]. Closed-form solutions were investigated [9, 10], but they do not allow overlap of constraints (the same parameters may not appear in different constraints). Because of that, many types of constraints can not be represented. The framework presented here tries to overcome such limitations.

2 Problem definition

A Bayesian network is a triple $(G, \mathcal{X}, \mathcal{P})$, where G is a directed acyclic graph with n nodes associated to discrete random variables $\mathcal{X}$ (a variable per node), and $\mathcal{P}$ is a collection of parameters $\theta_{ijk} = p(x_i^k | pa_i^j)$, with $\sum_k \theta_{ijk} = 1$, where x_i^k is a value or state of X_i and pa_i^j a complete instantiation for the parents PA_i of X_i in G (it represents a set of states for PA_i). In a BN every variable is conditionally independent of its non-descendants given its parents (Markov condition). Thus the joint probability distribution is obtained by $p(X_1, \ldots, X_n) = \prod_i p(X_i | PA_i)$. We focus on parameter learning in a BN where its structure is known in advance. Given a data set D where each element is a sample of the BN variables, the goal of parameter learning is to find the most probable values for the vector θ, which can be quantified by the log likelihood function $L_D(\theta) = \log(p(D|\theta))$. Assuming that samples are drawn independently from the underlying distribution, we need to maximize $L_D(\theta) = \log \prod_{ijk} \theta_{ijk}^{n_{ijk}}$, where n_{ijk} indicates how many elements of D contain both x_i^k and pa_i^j. Maximum likelihood estimation has its optimum at $\theta_{ijk} = \frac{n_{ijk}}{\sum_k n_{ijk}}$.

Another usual parameter learning technique is the Dirichlet model, where one starts by assuming that an expert has specified a prior BN, denoted by $\mathcal{N}_p$, that conveys her prior beliefs. The goal is to learn the parameters of multinomial distributions on θ_{ij} using both $\mathcal{N}_p$ and data. The Dirichlet distribution is a natural parametric model for $p(\theta_{ij})$, because it is conjugate with the multinomial distribution. A possible parametrization is $p(\theta_{ij}) \propto \prod_k \theta_{ijk}^{s\tau_{ijk}-1}$ for $s \geq 0$ and $\sum_k \tau_{ijk} = 1$, where the hyper-parameter s controls dispersion and hyper-parameters τ_{ijk} control location [17]. The parameter s is often interpreted as the *size* of a database encoding the same prior beliefs as the Dirichlet distribution. We assume that $\mathcal{N}_p$ is associated with a single positive number s that encodes the *quality* of the prior BN, and that parameters of $\mathcal{N}_p$ define τ such that τ_{ijk} corresponds to θ_{ijk} in the prior. Then, using expectation as estimator, the optimal estimate θ_{ijk} is the posterior expected value: $\theta_{ijk} = \frac{s\tau_{ijk}+n_{ijk}}{s+\sum_k n_{ijk}}$.

Standard estimation methods are usually enough when there are enough data. However, when a small amount of data is available, they may produce unreliable estimations. A way to improve estimations is through the use of constraints. Let θ_A be a sequence of parameters, α_A a corresponding sequence of constant numbers and α also a constant. A *linear relationship constraint* is defined as

$$\sum_{\theta_{ijk}\in\theta_A,\alpha_{ijk}\in\alpha_A} \alpha_{ijk} \cdot \theta_{ijk} \leq \alpha, \qquad (1)$$

that is, any linear constraint over parameters can be expressed as a *linear relationship constraint*. For instance, qualitative influences and synergies [18] can be expressed by linear constraints. Suppose X_1, X_2, X_3 are random variables that assume values in $\{x_i^1, x_i^2\}$, with x_i^2 greater than x_i^1, such that X_1 is the child of X_2 and X_3. It is said that X_2 has a positive influence on X_1 if $p(x_1^2|x_2^2, x_3^1) \geq p(x_1^2|x_2^1, x_3^1)$[1] and $p(x_1^2|x_2^2, x_3^2) \geq p(x_1^2|x_2^1, x_3^2)$, that is, a greater value of X_2 increases the probability of a greater value of X_1 (for both values of X_3). A positive synergy of two parents in a common child happens when the parents influence the child together, for example, $p(x_1^2|x_2^1, x_3^1) + p(x_1^2|x_2^2, x_3^2) \geq p(x_1^2|x_2^1, x_3^2) + p(x_1^2|x_2^2, x_3^1)$. This means that a greater value of the child is more likely when the parents have the same value. In fact these are just simple (but important) examples of constraints that are allowed. Other examples are sum of parameters, range, relationship, and ratio constraints [9], weak and strong, monotonic and non-monotonic influences and synergies [11], among many others. Our assumption about constraints is even more general: they must define a (possibly non-linear) convex parameter space, that is, any constraint in the form $h(\theta) \leq 0$, where h is convex, is allowed. Most of recent literature in this topic can be expressed by convex constraints [9, 10, 11, 18]. Such flexibility allows for a better description of the knowledge, as we have no restriction regarding the number of times a parameter appear in constraints or whether constraints involve distinct distributions of the BN.

[1]We use a probability notation for ease of expose, as each parameter θ_{ijk} is a probability value of the network.

3 Parameter learning using convex optimization

If we only have constraints on θ, ML can be solved by convex programming, as a maximization of a concave log-likelihood function must be solved. There are optimization algorithms to solve convex programming in polynomial time [2, 4]. They can be as fast as linear programming solvers. Convex programming has the attractive property that any local optimum is also a global optimum [4]. This idea of constrained ML is rather intuitive in its interpretation and in the method to solve it. However ML does not allow us to define prior assessments (probabilities cannot be directly taken as constraints because if they are fixed at this stage, the empirical data cannot change them). The interpretation of constraints only as *hard* constraints on estimations is eventually too inflexible, and in some cases it is more profitable to interpret expert's belief as a *partial assessment of a prior distribution*.

Besides constraints on parameters θ_{ijk}, we allow constraints on hyper-parameters τ_{ijk}. The idea is to work with both constraints on θ and constraints on τ. So, assume that an expert has specified two sets of constraints denoted by $\mathcal{C}_p$ and $\mathcal{C}$, that conveys her prior beliefs and some knowledge about parameters, respectively. The content of $\mathcal{C}_p$ is viewed as the set of *constraints on the hyper-parameters* τ_{ijk} of Dirichlet distributions for a fixed value of s, while $\mathcal{C}$ is the set of *hard* constraints on estimations. Constraints on $\mathcal{C}_p$ are *constraints on the prior*, not in the probability values themselves. The only assumption for constraints on $\mathcal{C}_p$ is that they must be convex constraints on τ (this is same assumption as for $\mathcal{C}$). If the expert is certain, she defines a constraint over θ. Otherwise, a similar constraint, but now over τ, may be used. In summary, constraints of $\mathcal{C}_p$ and $\mathcal{C}$ can be specified similarly. The former defines restrictions on parameters τ, while the latter defines restrictions for θ. This formulation is based on the *Imprecise Dirichlet Model* [17], which has received great attention recently [5, 14].

The result is a set of distributions that must satisfy

$\mathcal{C}(\theta)$, $\mathcal{C}_p(\tau)$ and equations

$$\theta_{ijk} = \frac{s\tau_{ijk} + n_{ijk}}{s + \sum_k n_{ijk}}, \tag{2}$$

where n_{ijk} are the counts from the data set (simplex constraints $\forall_{ij} \sum_k \theta_{ijk} = 1$ and $\forall_{ij} \sum_k \tau_{ijk} = 1$ are assumed to be in $\mathcal{C}(\theta)$ and $\mathcal{C}_p(\tau)$). Equation (2) is the *glue* between $\mathcal{C}(\theta)$ and $\mathcal{C}_p(\tau)$. This formulation has several attractive features. First, it deals with qualitative and numerical aspects in a uniform manner. Second, it uses constraints on priors and on estimations, making possible to the expert to state both *hard* and *soft* constraints. As any convex constraint is allowed, it is possible to specify precise probability measures as well as vacuous beliefs (the specification of a single prior is also possible as it is a sub-case). Third, a single hyper-parameter s must be elicited to capture the quality of the prior. Fourth, computations are efficient as all constraints are convex. Still, all these constraints define a set of distributions. Then we employ the maximum entropy principle, locally applied to each conditional distribution, to select one distribution from this set. Distributions of maximum entropy are conservative and tend to agree with frequencies [16]. So, the framework can be summarized as the following optimization problem:

$$\max_\theta - \sum_{ijk} \theta_{ijk} \log \theta_{ijk}, \tag{3}$$

subject to Equations (2), convex constraints $\mathcal{C}_p(\tau)$ and convex constraints $\mathcal{C}(\theta)$. This formulation can be polynomially solved by convex programming, as Equation (3) is the maximization of a concave function [8] subject to convex constraints. The resulting θ is our estimation.

Finally, we point out that Equations (2) tend the values of θ_{ijk} to be close to $\frac{n_{ijk}}{\sum_k n_{ijk}}$ (the frequencies of θ_{ijk} in the data set), while constraints on θ_{ijk} defined by the expert might (or might not) impose another different value. As we assume that hard constraints specified by the expert are correct, a penalty optimization variable is introduced in Equation (2) to guarantee that preference is given to the hard constraints defined by the expert and that the problem will not be infeasible (as long as expert's constraints are not already infeasible, in which case the expert should update her beliefs).

4 Experiments

We perform experiments using data sets with 10, 100 and 1000 samples. Three well–known networks are evaluated: Asia (Lauritzen and Spiegelhalter), Alarm (Beinlich et al.), and Insurance network (Binder et al.). For each network, we take one given parametrization as our true model and generate samples from it. Then Kullback–Leibler (KL) divergence is performed to measure the difference from distributions of estimated networks to distributions of true networks. Random linear (convex) constraints are generated from 1 to 5 parameters each. The constraints are created over the true network (so they are certainly correct) in number equal to the number of conditional distributions in the corresponding network. For each configuration, we work with 30 random sets of data and constraints. Averages of KL divergence are presented in Table 1. Columns have results of standard ML, Constrained ML and Constrained Maximum Entropy (CME).

Results indicate a strong decrease in the divergence when working with constraints (second and third columns of each block). Moreover, benefits are more significant with larger networks. We note that results with constraints using only 10 samples are better than results using 100 or even 1000 samples without constraints, which indicate a relevant decrease in the amount of data that would be necessary for achieving the same accuracy. The third column of each block shows results for the combination of constraints on priors and estimations using the maximum entropy idea. It is interesting to note the advantages when compared to the second column, because CML already uses constraints on estimations.

We also consider the problem of recognizing facial action units from real image data. Based on the Facial Action Coding System [6], facial behaviors can be decomposed into a set of Action Units (AUs). We work with a BN with 28 nodes to recognize 14 common occurring AUs (there are a hidden and a measurement node for each AU). The structure of the BN is learned as described in Tong et al. [13]. We define 42 simple linear constraints, mainly describing influences among AUs. The 8000 images from Cohn and Kanade's DFAT-504 database are used. Testing is performed over 20% of the data (not chosen for training). We consider training data sets with 100 and 1000 samples (as constraints are more relevant when insufficient data are available), chosen randomly from the training database. Results are shown in Table 2. CME obtains an overall recognition rate (percentage of correctly classified cases) of 93.1%, which is similar to current state-of-the-art results. For instance, Tong et at. [13] report 93.3%, Bartlett et al. [3] report 93.6%, and other methods have results with slight variance [12, 15].

5 Conclusion

This paper presents a framework for parameter learning when domain knowledge is available in the form

Network	Nodes	Distr	Dimension	10 samples			100 samples			1000 samples		
				ML	CML	CME	ML	CML	CME	ML	CML	CME
Asia	9	21	21	1.22	0.61	0.25	0.27	0.14	0.11	0.05	0.03	0.03
Alarm	37	243	509	3.26	1.80	0.61	2.51	1.19	0.47	1.12	0.58	0.26
Insurance	27	411	1008	3.62	1.56	0.63	2.24	0.92	0.44	0.96	0.40	0.20

Table 1. Average KL divergence using random samples and constraints.

Rate	100 samples		1000 samples	
	ML	CME	ML	CME
Positive	65.6%	75.7%	73.2%	83.9%
Negative	95.8%	97.7%	95.9%	97.0%

Table 2. Positive and negative rates for AU recognition.

of convex constraints. We have introduced a new idea based on the Imprecise Dirichlet Model and the maximum entropy criterion that is able to deal with constraints on priors and on estimations. The framework is fast and guarantees to find the global optimum solution. Through experiments with well-known networks, we show that both constraints on priors and estimations are important to improve parameter learning accuracy.

The main contribution of this work is to allow an expert to specify her knowledge using *hard* constraints on estimations and *soft* constraints on priors, with no restrictions on the format of constraints besides convexity. As far as we know, no previous methods were able to handle such general situation. The idea can also be embedded into an iterative procedure to treat incomplete data, similar to the Expectation-Maximization (EM) method. This discussion is left for the future, but we anticipate that the benefits are similar to those of complete data. We believe that the application of these ideas to real domains is promising and we intend to pursue that in a future work, as well as an investigation about the effect of wrong constraints.

Acknowledgments

This work is supported in part by a grant from the U.S. Army Research Office under grant number W911NF-06-1-0331.

References

[1] E. Altendorf, A. C. Restificar, and T. G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *UAI*, p. 18–26, 2005.

[2] E. D. Andersen, B. Jensen, R. Sandvik, and U. Worsoe. The improvements in mosek version 5. Technical report, Mosek Aps, 2007.

[3] M. S. Bartlett, G. C. Littlewort, M. G. Frank, C. Lainscsek, I. Fasel, J. R. Movellan. Automatic Recognition of Facial Actions in Spontaneous Expressions. *Journal of Multimedia*, 1(6):22–35, 2006.

[4] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series, 2001.

[5] C. P. de Campos and F. G. Cozman. Belief updating and learning in semi-qualitative probabilistic networks. In *UAI*, p. 153–160, 2005.

[6] P. Ekman and W. V. Friesen. *Facial action coding system: A technique for the measurement of facial movement*. Consulting Psychologists Press, 1978.

[7] A. Feelders. A new parameter learning method for bayesian networks with qualitative influences. In *UAI*, p. 117–124, 2007.

[8] E. H. Lieb. Some convexity and subadditivity properties of entropy. *B. of the American Math. Soc.*, 81(1), 1975.

[9] R. S. Niculescu. *Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks*. PhD thesis, Carnegie Mellon, 2005. CMU-CS-05-147.

[10] R. S. Niculescu, T. Mitchell, and B. Rao. Bayesian network learning with parameter constraints. *J. of Machine Learning Research*, 7(Jul):1357–1383, 2006.

[11] S. Renooij, L. C. van der Gaag, and S. Parsons. Context-specific sign-propagation in qualitative probabilistic networks. *Artif. Intell.*, 140(1-2):207–230, 2002.

[12] Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. *IEEE Trans. on PAMI*, 23(2):97–115, 2001.

[13] Y. Tong, W. Liao, and Q. Ji. Facial action unit recognition by exploiting their dynamic and semantic relationships. *IEEE Trans. on PAMI*, p. 1683–1699, 2007.

[14] L. Utkin and T. Augustin. Decision making under incomplete data using the imprecise dirichlet model. *Int. J. of Approximate Reasoning*, 44(3):322–338, 2007.

[15] M. F. Valstar, I. Patras, and M. Pantic. Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data. In *CVPR, W. Vision for Human-Computer Interaction*, 2005.

[16] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

[17] P. Walley. Inferences from multinomial data: Learning about a bag of marbles. *J. Royal Statistical Society B*, 58(1):3–57, 1996.

[18] M. P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artif. Intell.* 44:257–303, 1990.

Strategy Selection in Influence Diagrams using Imprecise Probabilities

Cassio P. de Campos
Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
decamc@rpi.edu

Qiang Ji
Electrical, Computer and Systems Eng. Dept.
Rensselaer Polytechnic Institute
Troy, NY, USA
jiq@rpi.edu

Abstract

This paper describes a new algorithm to solve the decision making problem in Influence Diagrams based on algorithms for credal networks. Decision nodes are associated to imprecise probability distributions and a reformulation is introduced that finds the global maximum strategy with respect to the expected utility. We work with Limited Memory Influence Diagrams, which generalize most Influence Diagram proposals and handle simultaneous decisions. Besides the global optimum method, we explore an anytime approximate solution with a guaranteed maximum error and show that imprecise probabilities are handled in a straightforward way. Complexity issues and experiments with random diagrams and an effects-based military planning problem are discussed.

1 INTRODUCTION

An influence diagram is a graphical model for decision making under uncertainty [13]. It is composed by a directed graph where utility nodes are associated to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represent actions to be taken. Given an influence diagram, a strategy defines which decision to take at each node, given the information available at that moment. Each strategy has a corresponding expected utility. One of the most important problems in influence diagrams is *strategy selection*, where we need to find the strategy with maximum expected utility. A simple approach is to evaluate each possible strategy and compare their expected utilities. However, the number of strategies grows exponentially in the number of decision to be taken.

In this paper, we propose a new idea to find the best strategy based on a reformulation of the problem as an inference in a credal network [4]. We show through experiments that this approach can handle small and medium diagrams exactly, and provides an anytime approximation in case we stop the process early. Our idea works with a very general class of influence diagrams, named *Limited Memory Influence Diagrams* (LIMIDs) [15]. *Limited Memory* means that the assumption of *no-forgetting* usually employed in Influence Diagrams (that is, values of observed variables and decisions that have been taken are remembered at all later times) is relaxed. This class of diagrams is interesting because most other influence diagram proposals can be efficiently converted into LIMIDs.

To solve strategy selection, many approaches work on special cases of influence diagrams, exploiting their characteristics to improve performance. In many cases, it is assumed that there is an ordering on which the decisions are to be taken and the no-forgetting rule, so as previous decisions are assumed to be known in the moment of the current decision [14, 18, 19, 20, 21]. The ordering of decision nodes is exploited to evaluate the optimal strategy. There are also proposals in the class of simultaneous influence diagrams, where decisions are assumed to have no antecedents. This assumption reduces the number of possible strategies and allows for factorization ideas [22]. LIMIDs do not have assumptions about no-forgetting and ordering for decisions, even though it is possible to convert diagrams that have such assumptions into LIMIDs.

In order to test our method, we generate a data set of random influence diagrams. Empirical results indicate that the accuracy of our method is better than other approaches'. We also apply our idea to solve an Effects-based operations (EBO) military planning. The EBO approach seeks for a campaign objective by considering direct, indirect and cascading effects of military, diplomatic, psychological and economic actions [6, 11]. We use an influence diagram to model an EBO hypothetical problem.

Section 2 introduces our notation for influence diagrams and the problem of strategy selection. Section 3 describes the framework of credal networks and the inference problem on such networks. Section 4 presents how we solve strategy selection through a reformulation of the problem as an inference in credal networks. Section 5 presents some experiments, including the EBO military planning problem, and finally Section 6 concludes the paper and indicates future work.

2 INFLUENCE DIAGRAMS

A Limited Memory Influence Diagram $\mathcal{I}$ is composed by a directed acyclic graph $(\mathcal{V}, E)$ where nodes are partitioned in three types: chance, decision and utility nodes. Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{U}$ be the set of chance, decision and utility nodes, respectively, and let $\mathcal{X} = \mathcal{C} \cup \mathcal{D}$. Links of E characterize dependencies among nodes. Explicitly, links toward a chance node indicate probabilistic dependence of the node on its parents; links toward a decision node indicate which information is available to take such decision, and links toward utility nodes represent that an utility for those parents is to be considered (utility nodes may not have children). Associated to each node, there are some parameters:

1. A *chance node* has an associated categorical random variable C with finite domain Ω_C and conditional probability distributions $p(C|\pi_j(C))$, for each configuration $\pi_j(C)$ of its parents $\pi(C)$ in the graph. j is used to indicate a configuration of the parents of C, that is, $\pi_j(C) \in \Omega_{\pi(C)}$, where the notation $\Omega_{\mathcal{V}'} = \times_{V \in \mathcal{V}'} \Omega_V$, for any $\mathcal{V}' \subseteq \mathcal{V}$.

2. A *decision node* D is associated to a finite set of mutually exclusive alternatives Ω_D. Parents of D describe the information that is available at the moment on which decision D has to be taken.

3. An *utility node* U is associated to a rational function $f_U : \Omega_{\pi(U)} \to \mathcal{Q}$. The value corresponding to a parent configuration is the profit (cost is viewed as negative profit) of such parent configuration. Utility nodes have no children.

A simple example is depicted in Figure 1. Decision nodes are represented by rectangles, chance nodes by ellipses and utility nodes by diamonds. *do_ground_attack* has an associated cost, which is depicted by the corresponding utility node. The same is modeled for *bomb_bridge*. The goal is to achieve *territory_occupation*, which also has an utility (the profit of the goal). *ground_attack* and *bridge_condition* represent the uncertain outcomes of the corresponding actions. Note that there is no known ordering on which decisions must be taken. Although decision nodes have no parents in this example, there is no such restriction.

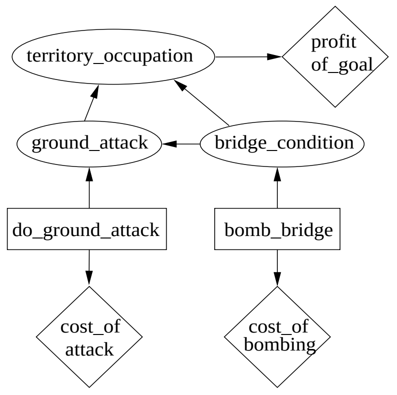


Figure 1: Simple Influence Diagram example.

A *policy* δ_D for the decision node D is a function $\delta_D : \Omega_{D \cup \pi(D)} \to [0, 1]$ defined for each alternative of D and each configuration of $\pi(D)$ such that, for each $\pi_j(D) \in \Omega_{\pi(D)}$ we have $\sum_{d \in \Omega_D} \delta_D(d, \pi_j(D)) = 1$. A *pure policy* is a policy such that its image is integer ($\delta_D : \Omega_{D \cup \pi(D)} \to \{0, 1\}$), and thus specifies with certainty which action (alternative of D) is taken for each parent configuration (in a pure policy, only one $\delta_D(d, \pi_j(D))$ for each $\pi_j(D)$ will be non-zero as they sum 1). A *strategy* Δ is a set of policies $\{\delta_D : D \in \mathcal{D}\}$, one for each decision node of the diagram. A *pure strategy* is composed only by pure policies.

The expected utility $EU(\Delta)$ of a strategy Δ is evaluated through the following equation:

$$\sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\prod_C p(x_C|\pi_j(C)) \prod_D \delta_D(x_D) \sum_U f_U(\pi_{j'}(U)) \right), \tag{1}$$

where x_C, $\pi_j(C)$, x_D and $\pi_{j'}(U)$ are respectively the projections of $\mathbf{x}$ in Ω_C, $\Omega_{\pi(C)}$, $\Omega_{D \cup \pi(D)}$ and $\Omega_{\pi(U)}$. This equation means that, given a strategy, its expected utility is the sum of the utility values weighted by the probability of each diagram configuration (for all configurations). The maximum expected utility is obtained over all possible strategies:

$$MEU = \max_{\Delta} EU(\Delta).$$

The problem of *strategy selection* is to obtain the strategy that maximizes its expected utility, that is, argmax $\max_\Delta EU(\Delta)$.

3 CREDAL NETWORKS

We need some concepts of credal networks before presenting the reformulation to solve strategy selection. A convex set of probability distributions is called a *credal set* [4]. A credal set for X is denoted by $K(X)$; we assume that every random variable is categorical and that every credal set has a finite number of vertices. Given a credal set $K(X)$ and an event A, the *upper* and *lower* probability of A are respectively $\max_{p(X)\in K(X)} p(A)$ and $\min_{p(X)\in K(X)} p(A)$. A conditional credal set is a set of conditional distributions, obtained by applying Bayes rule to each distribution in a credal set of joint distributions.

A (separately specified) *credal network* $N = (G, \mathbb{X}, \mathbb{K})$ is composed by a directed acyclic graph $G = (V, E)$ where each node of V is associated with a random variable $X_i \in \mathbb{X}$ and with a collection of conditional credal sets $K(X_i|\pi(X_i)) \in \mathbb{K}$, where $\pi(X_i)$ denotes the parents of X_i in the graph. Note that we have a conditional credal set related to X_i for each configuration $\pi_j(X_i) \in \Omega_{\pi(X_i)}$. A root node is associated with a single marginal credal set. We take that in a credal network every random variable is independent of its non-descendants non-parents given its parents; this is the *Markov condition* on the network. In this paper we adopt the concept of *strong independence*[1]: two random variables X_i and X_j are strongly independent when every extreme point of $K(X_i, X_j)$ satisfies standard stochastic independence of X_i and X_j (that is, $p(X_i|X_j) = p(X_i)$ and $p(X_j|X_i) = p(X_j)$) [4]. Strong independence is the most commonly adopted concept of independence for credal sets, probably due to its connection with standard stochastic independence.

Given a credal network, its *extension* is any joint credal set that satisfies all constraints encoded in the network. The *strong extension* $\mathcal{K}$ of a credal network is the largest joint credal set such that every variable is strongly independent of its non-descendants non-parents given its parents. The strong extension of a credal network is the joint credal set that contains every possible combination of vertices for all credal sets in the network [5]; that is, each vertex of a strong extension factorizes as follows:

$$p(X_1, \ldots, X_n) = \prod_i p(X_i|\pi(X_i)) \,. \qquad (2)$$

Thus, a credal network can be viewed as a representation for a set of Bayesian networks with distinct parameters but sharing the same graph.

3.1 INFERENCE

A *marginal inference* in a credal network is the computation of upper (or lower) probabilities in an extension of the network. If X_q is a *query* variable, then a marginal inference is the computation of tight bounds for $p(x_q)$ for one or more categories x_q of X_q. For inferences in strong extensions, it is known that distributions that maximize $p(x_q)$ belong to the set of vertices of the extension [12]. So, an inference can be produced by combinatorial optimization, as we must find a vertex for each local credal set $K(X_i|\pi(X_i))$ so that Expression (2) leads to a maximum of $p(x_q)$. In general, inference offers tremendous computational challenges, and exact inference algorithms based on enumeration of all potential vertices face serious difficulties [4].

A different way to solve the problem is to recognize that an upper (or lower) value for $p(x_q)$ may be obtained by the optimization of a multilinear polynomial over probability values, subject to constraints. This idea is discussed in the literature and different methods to reformulate the inference problem were proposed [7, 9]. Empirical results suggest that this is the most effective way for exact inferences. In the next section, we describe an idea based on bilinear programming [9] to perform inferences in credal networks and show how it can be employed to solve the strategy selection problem of influence diagrams.

4 STRATEGY SELECTION AS A CREDAL NET INFERENCE

Suppose we want to find the strategy Δ_{opt} that maximizes the expected utility in an influence diagram $\mathcal{I}$, that is, $\Delta_{opt} = \operatorname{argmax} MEU$. Let $\underline{f}$ and $\overline{f}$ be the minimum and maximum utility values specified in the diagram for all possible utility nodes and parent configurations, that is,

$$\underline{f} = \min_{U,\pi_j(U)} f_U(\pi_j(U)), \qquad \overline{f} = \max_{U,\pi_j(U)} f_U(\pi_j(U)).$$

We create an identical influence diagram $\mathcal{I}'$ except that the utility function f'_U (for each node U) is defined as

$$\forall \pi_j(U) \quad f'_U(\pi_j(U)) = \frac{f_U(\pi_j(U)) - \underline{f}}{\overline{f} - \underline{f}}.$$

The denominator is positive because $\underline{f} < \overline{f}$ (if $\underline{f} = \overline{f}$, then the influence diagram is trivial as all utility values are equal). We note that this transformation is similar to that proposed by Cooper [2]. It is not hard to see that $\operatorname{argmax} MEU = \operatorname{argmax} MEU'$ (just take the terms out of summations in Equation (1)), and

$$\max_\Delta EU'(\Delta) = \frac{\max_\Delta EU(\Delta) - |\mathcal{U}|\underline{f}}{\overline{f} - \underline{f}}.$$

This implies that strategy selection in $\mathcal{I}$ is the same as strategy selection in $\mathcal{I}'$. Now, we translate the selection problem of $\mathcal{I}'$ to a credal network inference. Suppose we define a credal network with a similar graph as $\mathcal{I}'$ such that:

[1]We note that other concepts of independence are found in the literature [3, 10].

- Chance nodes are directly translated as nodes of the credal network (parents are the same as in $\mathcal{I}'$).
- Utility nodes are translated to binary random nodes. Let U be an utility node with function f_U. In the credal network, U becomes a binary node (with the same parents as before) and categories u and $\neg u$ such that: $p(u|\pi_j(U)) = f_U(\pi_j(U))$ and $p(\neg u|\pi_j(U)) = 1 - p(u|\pi_j(U))$ [2].
- Decision nodes are translated to probabilistic nodes with imprecise distributions such that policies become probability distributions (in fact, according to our definition of policy, they are already greater than zero and sum 1). Thus, $p(d|\pi_j(D)) = \delta_D(d, \pi_j(D))$ for all d and $\pi_j(D)$. Note that $p(D|\pi_j(D))$, for each $\pi_j(D)$, is a distribution with unknown probability values (this interpretation of decision nodes as imprecise probability nodes is discussed by Antonucci and Zaffalon, see e.g. [1]).

Using this credal network formulation, the expected utility of a strategy Δ can be written as

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_{\mathcal{X}}} \left(\prod_X p_\Delta(x|\pi_j(X)) \sum_U p(u|\pi_{j'}(U)) \right),$$

where x, $\pi_j(X)$ and $\pi_{j'}(U)$ are projections of $\mathbf{x}$ into the corresponding domains, X ranges on all nodes corresponding to chance and decision nodes of the influence diagram, and p_Δ represents the distribution induced by the strategy Δ, that is, when the strategy is chosen, p_Δ is a known probability distribution.

With some simple manipulations, we have:

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_X} \left(p_\Delta(\mathbf{x}) \sum_U p(u|\pi_{j'}(U)) \right),$$

$$EU'(\Delta) = \sum_{\mathbf{x}\in\Omega_X} \left(\sum_U p(u|\pi_{j'}(U)) p_\Delta(\mathbf{x}) \right),$$

$$EU'(\Delta) = \sum_U \sum_{\mathbf{x}\in\Omega_X} p_\Delta(u, \mathbf{x}) = \sum_U p_\Delta(u),$$

and then

$$MEU' = \max_\Delta \sum_U p_\Delta(u) = \max_{p\in\mathcal{K}} \sum_U p(u),$$

where $p \in \mathcal{K}$ means that we select a distribution p in the extension of the credal network. In fact the only places p may vary are related to the imprecise probabilities of the former decision nodes. When we select p, we get a precise distribution that has a corresponding strategy Δ. So, we have a credal network and need to find a distribution p that maximizes the sum of marginal probabilities of the U nodes.

4.1 INFERENCE AS AN OPTIMIZATION PROBLEM

The sum of marginal inferences in the credal network can be formulated as a multilinear programming problem. The goal is to maximize the expression

$$\sum_U p(u) = \sum_U \sum_{\mathbf{x}\in\Omega_{\mathcal{X}}} \left(p(u|\pi_{j'}(U)) \prod_X p(x|\pi_j(X)) \right), \tag{3}$$

where x, $\pi_{j'}(U)$ and $\pi_j(X)$ are the projections of $\mathbf{x}$ in the corresponding domains, and where some distributions $p(X|\pi_j(X))$ are precisely known and others are imprecise. In this formulation we must deal with a large number of multilinear terms. To avoid them, we briefly describe the bilinear transformation procedure proposed by de Campos and Cozman [9] to replace the large Expression (3) by simple bilinear expressions. We refer to [9] for additional details.

The idea is based on a *precedence ordering* of the network variables, which is an ordering where all ancestors of a given variable in the network's graph appear before it in the ordering. The bilinear transformation algorithm processes the network variables top-down: at each step some constraints are generated that define the relationship between the query and the current variable being processed. A variable may be processed only if all its ancestors have already been processed. The active nodes at each step form a path-decomposition of the network's graph.

To better explain the method, we take the example of Figure 1. For simplicity, assume that variables are binary[2] (with categories b and $\neg b$) renamed as follows: *do_ground_attack* is D_1, *bomb_bridge* is D_2, *cost_of_attack* is U_1, *cost_of_bombing* is U_2, *ground_attack* is C_1, *bridge_condition* is C_2, *territory_occupation* is C_3, and finally *profit_of_goal* is U_3.

After the translation of the utility functions into probability distributions and the replacement of decision nodes by nodes with imprecise probabilities (as previously described), we have a credal network and need to maximize the sum of the marginal probabilities of the U nodes. In fact this is an extension of the standard query in a credal network, because we have a summation instead of a single probability to maximize. So the objective function is max $p(u_1) + p(u_2) + p(u_3)$ (there are three utility nodes in the example) subject to constraints that define each marginal probability $p(u_1)$, $p(u_2)$ and $p(u_3)$. To create these constraints, we run a symbolic inference based on the precedence ordering for each of the marginal probabilities. The constraints for $p(u_1)$ and $p(u_2)$ are very

[2]The method works on non-binary variables as well. The assumption is made here for ease of expose.

simple: $p(u_1) = p(u_1|d_1)p(d_1) + p(u_1|\neg d_1)p(\neg d_1)$ and $p(u_2) = p(u_2|d_2)p(d_2)+p(u_2|\neg d_2)p(\neg d_2)$, because they only depend on one other variable. Note that $p(d_1)$, $p(\neg d_1)$, $p(d_2)$, and $p(\neg d_2)$ that appear in these constraints are unknown and thus become optimization variables in the bilinear problem.

To write the constraints for $p(u_3)$, we need to choose a precedence ordering. We will use the ordering $D_2, C_2, D_1, C_1, C_3, U_3$ (variables U_1 and U_2 do not appear in the order as they are not relevant to evaluate the marginal $p(u_3)$). Hence, the first variable to be processed is D_2. We write a constraint that relates the query u_3 and probabilities $p(D_2)$ (which are defined in the network specification):

$$p(u_3) = \sum_{d \in \{d_2, \neg d_2\}} p(d) \cdot p(u_3|d).$$

D_2 now appears in the conditional part of $p(u_3|d)$, which may be viewed as an artificial term in the optimization, as it does not appear in the network. Because of that, we must create constraints to define $p(u_3|d)$ in terms of network parameters (for all categories $d \in D_2$). According to our chosen ordering, the current variable to be processed is C_2. Thus,

$$\begin{aligned} p(u_3|d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|d_2) \cdot p(u_3|c), \\ p(u_3|\neg d_2) &= \sum_{c \in \{c_2, \neg c_2\}} p(c|\neg d_2) \cdot p(u_3|c). \end{aligned}$$

Note that $p(u_3|c) = p(u_3|c, d)$ (for any d), so we use the simpler. At this stage, our query is conditioned on C_2. Following the same idea, we process D_1, obtaining

$$\begin{aligned} p(u_3|c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|c_2, d), \\ p(u_3|\neg c_2) &= \sum_{d \in \{d_1, \neg d_1\}} p(d) \cdot p(u_3|\neg c_2, d). \end{aligned}$$

Now the current variable to be treated is C_1, and our query is conditioned on C_2, D_1, that is, we must define how to evaluate $p(u_3|C_2, D_1)$ for all configurations. Thus, for all $c \in \{c_2, \neg c_2\}$ and $d \in \{d_1, \neg d_1\}$:

$$p(u_3|c, d) = \sum_{c' \in \{c_1, \neg c_1\}} p(c'|c, d) \cdot p(u_3|c, c').$$

At this moment, u_3 is conditioned on C_1, C_2 in the artificial term $p(u_3|c, c')$ (D_1 is not present in the artificial term as C_1, C_2 separate u_3 from D_1). Now we process C_3: for all $c' \in \{c_1, \neg c_1\}$ and $c \in \{c_2, \neg c_2\}$

$$p(u_3|c, c') = \sum_{c'' \in \{c_3, \neg c_3\}} p(c''|c, c') \cdot p(u_3|c'').$$

Note that, as $p(u_3|c'')$ is specified in the network, we can stop. All artificial terms are related (through constraints) to parameters of the network. Besides all these constraints, we also include simplex constraints to ensure that probabilities sum 1.

Hence, we have a collection of linear and bilinear constraints on which non-linear programming can be employed [7]. It is also possible to use linear integer programming [9]. The steps to achieve a linear integer programming formulation are simple, because the only non-linear terms of the problem have the format $b \cdot t$, where $b \in \{0, 1\}$ and $t \in [0, 1]$. b is an unknown probability value of the credal network (which is zero or one because the solution we look for lies on extreme points of credal sets [12]) and t is a constant or an artificial term created in the procedure just described. To *linearize* the problem, $b \cdot t$ is replaced by an additional artificial optimization variable y and the following constraints are inserted: $0 \leq y \leq b$ and $t - 1 + b \leq y \leq t$. After replacing all non-linear terms using this idea, the problem becomes a linear integer programming problem, where a solution is also a solution for the strategy selection in the initial influence diagram.

We emphasize that, as we are translating the strategy selection problem into a credal network inference, it is straightforward to use imprecise probabilities in the chance nodes of the influence diagram. Intervals or sets of probabilities may be used. The translation works in the same way, but the generated problem will have more imprecise probabilities to optimize.

The following theorem shows that, when reformulating the strategy selection problem as a modified credal network inference, we are not making use of "more effort" than necessary, that is, strategy selection has the same complexity as inference in credal networks.

Theorem 1 *Let $\mathcal{I}$ be a LIMID and k a rational. Deciding whether there is a strategy Δ such that MEU is greater than k is NP-Complete when $\mathcal{I}$ has bounded induced width,*[3] *and NP^{PP}-Complete in general.*

Proof sketch: Pertinence for the bounded induced width case is achieved because (given a strategy) we can compute *MEU* and verify if it is greater than k in polynomial time (using the reformulation and the sum of marginal queries, each marginal query takes polynomial time in a bounded induced width Bayesian network); in the general case, we can perform this verification using a PP oracle. Hardness for the bounded induced width case is obtained with the same reduc-

[3]The maximum clique and the maximum degree in the moral graph are bounded by a logarithmic function in the size of the input needed to specify the problem, which for instance includes polytrees.

tion as in [8] from the MAXSAT problem (replacing the credal nodes with decision nodes and introducing a single utility node). In the general case, the same reduction as in [17] from E-MAJSAT can be used (MAP nodes are replaced by decision nodes). □

5 EXPERIMENTS

We conduct two experiments with the procedure. First, we use random generated influence diagrams to compare the solutions obtained by our procedure (which we call CR for *credal reformulation*) against the *Single Policy Updating* (SPU) of Lauritzen and Nilsson [15]. Later we work with a practical EBO military planning problem and compare the method against the factorization of Zhang and Ji [22].[4]

Concerning random influence diagrams, we have generated a data set based on the total number of nodes and the number of decision nodes. The configurations chosen are presented in the first two columns of Table 1. We have from 10 to 120 nodes, where 3 to 35 are decision nodes. The number of utility nodes is chosen equal to the number of decision nodes. Each line in Table 1 contains the average result for 30 random generated diagrams within that configuration. The third column of the table shows the approximate average number of distinct strategies in the diagrams that would need to be evaluated by a brute force method.

The three columns of the CR method show the time spent to solve the problem, the number of nodes evaluated in the branch-and-bound tree of the optimization procedure (which is significantly smaller than the total number of strategies in brute force) and the maximum error of the solution (all numbers are averages). After the reformulation, the CPLEX solver [16] is used, which includes a heuristic search before starting the branch-and-bound procedure. The evaluations of this heuristic search are not counted in the fifth column of Table 1. Note that the first five rows are separated from the last three because they strongly differ on the size of the search space (exact solutions were found only for the former). The maximum error of each solution is obtained straightforward from the relaxation of the linear integer problem. The last two columns of Table 1 show the time and maximum error of the SPU approximate procedure. Although very fast, the SPU procedure has worse accuracy than the "approximate" CR (solution was approximate in last three rows because we have imposed a time-limit of ten minutes for each run). Furthermore, SPU does not provide an upper bound for the best possible expected utility, as obtained by CR. Still, a possible improvement is to use SPU to provide an initial guess to the optimization.

[4]The factorization idea only works on simultaneous influence diagrams, so it was not used in the other test cases.

5.1 EBO MILITARY PLANNING

In this section we describe the performance of our method in an hypothetical Effects-based Operations planning problem [11]. An influence diagram similar to the model described by Zhang and Ji [22] is employed. Its graph is shown in Figure 2. The goal is to win a war, which is represented by the *Hypothesis* node (on top of Figure 2). Just below there are the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, which are directly related to the main goal. There are eleven decision nodes (represented by rectangles): *destroy_C2* (C2 stands for *Command and Control*), *destroy_Radars*, *destroy_Communications*, *launch_air_strike*, *destroy_RD*, *destroy_storage*, *destroy_assembly*, *launch_ground_attack*, *launch_broadcasting*, *capture_bodyguard*, *use_special_force*. Just above decision nodes, we have chance nodes representing the outcomes of performing such actions (they indicate the workability of such systems), and below we have utility nodes (diamond-shaped nodes) describing the cost of each action. Furthermore, we have six chance nodes (in the center of the figure) indicating general workability of *IADS* (Integrated Air Defense System), *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody* with respect to enemy forces. The overall profit of winning is given by the node U_H, child of *Hypothesis*.

As this is an hypothetical example, we define utility functions and probability distributions as follows:

- Probability of *Hypothesis* is one given that all subgoals are achieved. If one of subgoals is not achieved, then the probability of *Hypothesis* is 60%; if two of them are not achieved, then the probability of success is 30%; if none of subgoals is achieved, then we certainly fail in the campaign.

- For the subgoals *Air_superiority*, *Territory_occupation*, and *Commander_surrender*, we define that the subgoal is accomplished with probability one when both children were achieved, 50% when only one child is achieved, and zero when none is achieved.

- For the probabilities of *IADS*, *Air_force*, *Artillery*, *Ground_force*, *Morale* and *Commander_in_custody*, we define a decrease of 50% for each unaccomplished child (with a minimum of zero, of course). Any node has probability zero if two or more of its children are not achieved.

- The outcomes of actions (chance nodes above decision nodes) have 90% of success. For exam-

Nodes		Approx.# of	CR			SPU	
Total	Decision	Strategies	Time(sec)	Evals (B&B)	Max.Error(%)	Time(sec)	Max.Error(%)
10	3	2^{17}	0.66	5	0.000	0.10	0.740
20	6	2^{34}	1.73	125	0.000	0.39	2.788
50	10	2^{51}	30.42	4048	0.000	1.62	2.837
60	15	2^{52}	29.77	2937	0.000	2.99	1.964
70	20	2^{54}	125.06	7132	0.000	5.52	3.448
120	25	2^{102}	254.80	15626	0.544	11.58	2.193
120	30	2^{116}	403.13	5617	4.639	13.79	7.281
120	35	2^{120}	578.99	9307	5.983	16.87	11.584

Table 1: Average results on 30 random influence diagrams of different sizes for the CR and SPU methods.

ple, *destroy_Radars* will have *EW/GCI_radars* destroyed with 90% of odds (EW/GCI means *Early Warning/Ground Control Interception*).

- The reward of achieving the main goal is 1000, while not achieving it costs 500.
- Costs of actions are as follows: *ground_attack* is 150, *use_special_force* is 100, *capture_bodyguard* is 80, *air_strike* is 50, and other actions cost 20 each.

For this problem, the best strategy found by SPU has expected utility of −55.2825, and suggests to take all action except *destroy_RD*, *destroy_storage*, *destroy_assembly* and *launch_ground_attack*. The global optimum strategy is found in less than 5 seconds with our method and has expected utility equal to 156.4051 (all actions are taken). This is much faster than the solution reported by [22] (around 45 seconds).

6 CONCLUSION

We discuss in this paper a new idea for strategy selection in Influence Diagrams. We work with the Limited Memory Influence Diagram, as it generalizes many of the influence diagram proposals. The main contribution is the reformulation of the problem as a credal network inference, which makes possible to find the global maximum strategy for small- and medium-sized influence diagrams. Experiments indicate that many instances can be treated exactly. As far as we know, no deep investigation of exact procedures for this class of diagrams has been conducted.

Because of the characteristics of our procedure, an anytime approximate solution with a maximum guaranteed error is available during computations. It is clear that large diagrams must be treated approximately. Nevertheless, in the conducted experiments, our method produced results that surpass existing algorithms. Although spending more time, many situations require a solution to be as good as possible, while time is a secondary issue. The ability of our approach to provide an upper bound for the result is also valuable, which is not available with the SPU method.

We also discuss the theoretical complexity of the problem, which is derived from the known properties of MAP problems in Bayesian networks and belief updating inferences in credal networks. The complexity results show that the proposed idea is not making use of a harder problem to solve a simpler one, as the complexity of strategy selection is the same as the complexity of inferences in credal networks.

Because strategy selection in influence diagrams and inferences in credal networks are related, improvements on algorithms of credal networks can be directly applied to influence diagram problems. The application of other approximate techniques based on credal networks seems a natural path for investigation. We also intend to explore other optimization criteria for influence diagrams with imprecise probabilities, besides expected utility. Proposals in the theory of imprecise probabilities might be applied to this setting.

Acknowledgements

The work described in this paper is supported in part by the U.S. Army Research Office grant W911NF0610331.

References

[1] A. Antonucci and M. Zaffalon. Decision-theoretic specification of credal networks: A unified language for uncertain modeling with sets of Bayesian networks. *Int. J. Approx. Reason.*, in press, doi:10.1016/j.ijar.2008.02.005, 2008.

[2] G. F. Cooper. A method for using belief updating as influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 55–63, Minneapolis, 1988.

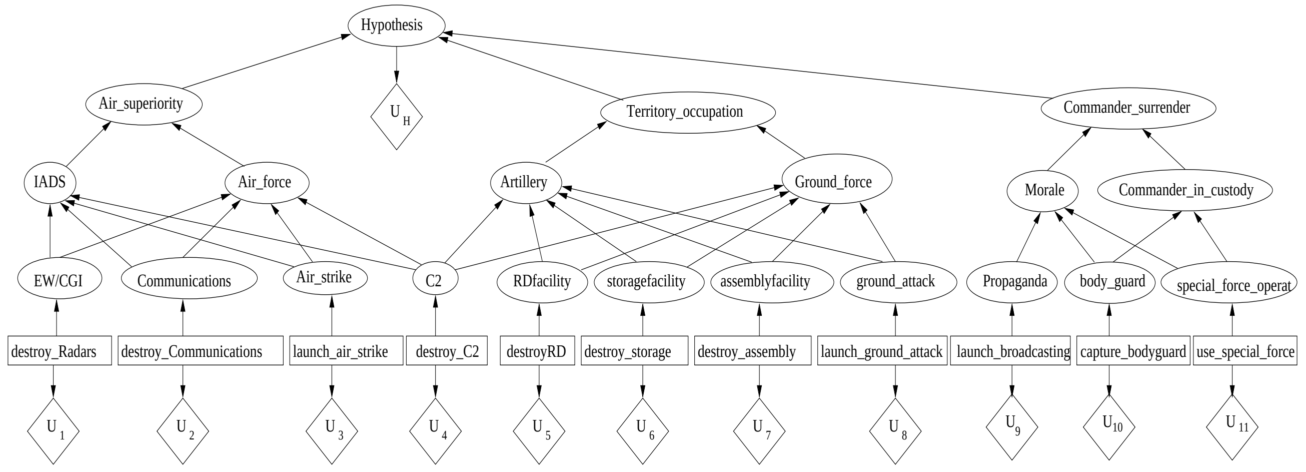


Figure 2: Influence Diagram for an hypothetical EBO-based planning problem.

[3] I. Couso, S. Moral, and P. Walley. A survey of concepts of independence for imprecise probabilities. *Risk, Decision and Policy*, 5:165–181, 2000.

[4] F. G. Cozman. Credal networks. *Artif. Intelligence*, 120:199–233, 2000.

[5] F. G. Cozman. Separation properties of sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 107–115, San Francisco, 2000.

[6] P. Davis. Effects-based operations: a grand challenge for the analytical community. Technical report, Rand corp., 2003. MR1477.

[7] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *Second Starting AI Researcher Symposium*, p. 50–61, Valencia, 2004. IOS Press.

[8] C. P. de Campos and F. G. Cozman. The inferential complexity of Bayesian and credal networks. In *Int. Joint Conf. on Artif. Intelligence*, p. 1313–1318, 2005.

[9] C. P. de Campos and F. G. Cozman. Inference in credal networks through integer programming. In *Int. Symp. on Imprecise Probability: Theories and Applications*, p. 145–154, 2007.

[10] L. de Campos and S. Moral. Independence concepts for convex sets of probabilities. In *Conf. on Uncertainty in Artif. Intelligence*, p. 108–115, San Francisco, 1995.

[11] D. A. Deptula. Effects-based operations: change in the nature of warfare. *Defense and Airpower Series*, p. 3–6, 2001.

[12] E. Fagiuoli and M. Zaffalon. 2U: An exact interval propagation algorithm for polytrees with binary variables. *Artif. Intelligence*, 106(1):77–107, 1998.

[13] R. A. Howard and J. E. Matheson. *Influence diagrams*, volume II, p. 719–762. Strategic Decisions Group, Menlo Park, 1984.

[14] F. Jensen, F. V. Jensen, and S. L. Dittmer. From influence diagrams to junction trees. In *Conf. on Uncertainty in Artif. Intelligence*, p. 367–373, San Francisco, 1994.

[15] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1238–1251, 2001.

[16] Ilog Optimization. Cplex documentation. http://www.ilog.com, 1990.

[17] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artif. Intelligence Research*, 21:101–133, 2004.

[18] R. Qi and D. Poole. A new method for influence diagram evaluation. *Computational Intelligence*, 11:1:1–34, 1995.

[19] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986.

[20] N. L. Zhang. Probabilistic inferences in influence diagrams. In *Conf. on Uncertainty in Artif. Intelligence*, p. 514–522, Madison, 1998.

[21] N. L. Zhang and D. Poole. Stepwise-decomposable influence diagram. In *Int. Conf. on Principles of Knowledge Representation and Reasoning*, p. 141–152, Cambridge, 1992.

[22] W. Zhang and Q. Ji. A factorization approach to evaluating simultaneous influence diagrams. *IEEE Transactions on Systems, Man and Cybernetics A*, 36(4):746–757, 2006.

Structure Learning of Bayesian Networks using Constraints

Cassio P. de Campos CASSIO@IDSIA.CH
Dalle Molle Institute for Artificial Intelligence (IDSIA), Galleria 2, Manno 6928, Switzerland

Zhi Zeng ZENGZ@RPI.EDU
Qiang Ji JIQ@RPI.EDU
Rensselaer Polytechnic Institute (RPI), 110 8th St., Troy NY 12180, USA

Abstract

This paper addresses exact learning of Bayesian network structure from data and expert's knowledge based on score functions that are decomposable. First, it describes useful properties that strongly reduce the time and memory costs of many known methods such as hill-climbing, dynamic programming and sampling variable orderings. Secondly, a branch and bound algorithm is presented that integrates parameter and structural constraints with data in a way to guarantee global optimality with respect to the score function. It is an any-time procedure because, if stopped, it provides the best current solution and an estimation about how far it is from the global solution. We show empirically the advantages of the properties and the constraints, and the applicability of the algorithm to large data sets (up to one hundred variables) that cannot be handled by other current methods (limited to around 30 variables).

1. Introduction

A Bayesian network (BN) is a probabilistic graphical model that relies on a structured dependency among random variables to represent a joint probability distribution in a compact and efficient manner. It is composed by a directed acyclic graph (DAG) where nodes are associated to random variables and conditional probability distributions are defined for variables given their parents in the graph. Learning the graph (or structure) of a BN from data is one of the most challenging problems in such models. Best exact known methods take exponential time on the number of variables and are applicable to small settings (around 30 variables). Approximate procedures can handle larger networks, but usually they get stuck in local maxima. Nevertheless, the quality of the structure plays a crucial role in the accuracy of the model. If the dependency among variables is not properly learned, the estimated distribution may be far from the *correct* one. In general terms, the problem is to find the best structure (DAG) according to some score function that depends on the data (Heckerman et al., 1995). There are other approaches to learn a structure that are not based on scoring (for example taking some statistical similarity among variables), but we do not discuss them in this paper. The research on this topic is active, e.g. (Chickering, 2002; Teyssier & Koller, 2005; Tsamardinos et al., 2006). Best exact ideas (where it is guaranteed to find the global best scoring structure) are based on dynamic programming (Koivisto et al., 2004; Singh & Moore, 2005; Koivisto, 2006; Silander & Myllymaki, 2006), and they spend time and memory proportional to $n \cdot 2^n$, where n is the number of variables. Such complexity forbids the use of those methods to a couple of tens of variables, mostly because of memory consumption.

In the first part of this paper, we present some properties of the problem that bring a considerable improvement on many known methods. We perform the analysis over some well known criteria: *Akaike Information Criterion* (AIC), and the *Minimum Description Length* (MDL), which is equivalent to the *Bayesian Information Criterion* (BIC). However, results extrapolate to the Bayesian Dirichlet (BD) scoring (Cooper & Herskovits, 1992) and some derivations under a few assumptions. We show that the search space of possible structures can be reduced drastically without losing the global optimality guarantee and that the memory requirements are very small in many practical cases

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

(we show empirically that only a few thousand scores are stored for a problem with 50 variables and one thousand instances).

As data sets with many variables cannot be efficiently handled (unless P=NP, as the problem is known to be NP-hard (Chickering et al., 2003)), a desired property of a method is to produce an *any-time* solution, that is, the procedure, if stopped at any moment, provides an approximate solution, while if run until it finishes, a global optimum solution is found. However, the most efficient exact methods are not *any-time*. We propose a new any-time exact algorithm using a branch-and-bound (B&B) approach with caches. Scores are computed during the initialization and a poll is built. Then we perform the search over the possible graphs iterating over arcs. Although iterating over orderings is probably faster, iterating over arcs allows us to work with constraints in a straightforward way. Because of the B&B properties, the algorithm can be stopped at any-time with a best current solution found so far and an upper bound to the global optimum, which gives a kind of certificate to the answer and allows the user to stop the computation when she believes that the current solution is good enough. (Suzuki, 1996) has proposed a B&B method, but it is not a global exact algorithm, instead the search is conducted after a node ordering is fixed. Our method does not rely on a predefined ordering and finds a global optimum structure considering all possible orderings.

2. Bayesian networks

A BN represents a single joint probability density over a collection of random variables. It can be defined as a triple $(\mathcal{G}, \mathcal{X}, \mathcal{P})$, where $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ is a DAG with $V_\mathcal{G}$ a collection of n nodes associated to random variables $\mathcal{X}$ (a node per variable), and $E_\mathcal{G}$ a collection of arcs; $\mathcal{P}$ is a collection of conditional probability densities $p(X_i|PA_i)$ where PA_i denotes the parents of X_i in the graph (PA_i may be empty), respecting the relations of $E_\mathcal{G}$. We assume throughout that variables are categorical. In a BN every variable is conditionally independent of its non-descendants given its parents (Markov condition). This structure induces a joint probability distribution by the expression $p(X_1, \ldots, X_n) = \prod_i p(X_i|PA_i)$. Before proceeding, we define some notations. Let $r_i \geq 2$ be the number of discrete categories of X_i, q_i the number of elements in Ω_{PA_i} (the number of configurations of the parent set, that is, $q_i = \prod_{X_t \in PA_i} r_t$) and θ be the entire vector of parameters such as $\theta_{ijk} = p(x_i^k|pa_i^j)$, where $i \in \{1, \ldots, n\}$, $j \in \{1, ..., q_i\}$, $k \in \{1, ..., r_i\}$ (hence $x_i^k \in \Omega_{X_i}$ and $pa_i^j \in \Omega_{PA_i}$).

Given a complete data set $D = \{D_1, \ldots, D_N\}$ of with N instances, with $D_t = \{x_{1,t}^{k_1}, \ldots, x_{n,t}^{k_n}\}$ a instance of all variables, the goal of structure learning is to find a $\mathcal{G}$ that maximizes a score function such as MDL or AIC.

$$\max_\mathcal{G} s_D(\mathcal{G}) = \max_\theta (L_D(\theta) - t \cdot W),$$

where θ represents all parameters of the model (and thus depends on the graph $\mathcal{G}$), $t = \sum_{i=1}^n (q_i \cdot (r_i - 1))$ is the number of free parameters, W is criterion-specific ($W = \frac{\log N}{2}$ in MDL and $W = 1$ in AIC), and L_D is the log-likelihood function:

$$L_D(\theta) = \log \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}}, \qquad (1)$$

where n_{ijk} indicates how many elements of D contain both x_i^k and pa_i^j. This function can be written as $L_D(\theta) = \sum_{i=1}^n L_{D,i}(\theta_i)$, where $L_{D,i}(\theta_i) = \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} n_{ijk} \log \theta_{ijk}$. From now on, the subscript D is omitted for simplicity.

An important property of such criteria is that they are decomposable, that is, they can be applied to each node X_i separately: $\max_\mathcal{G} s(\mathcal{G}) = \max_\mathcal{G} \sum_{i=1}^n s_i(PA_i)$, where $s_i(PA_i) = L_i(PA_i) - t_i(PA_i) \cdot W$, with $L_i(PA_i) = \max_{\theta_i} L_i(\theta_i)$ (θ_i is the parameter vector related to X_i, so it depends on the choice of PA_i), and $t_i(PA_i) = q_i \cdot (r_i - 1)$. Because of this property and to avoid computing such functions several times, we create a cache that contains $s_i(PA_i)$ for each X_i and each parent set PA_i. Note that this cache may have an exponential size on n, as there are 2^{n-1} subsets of $\{X_1, \ldots, X_n\} \setminus \{X_i\}$ to be considered as parent sets. This gives a total space and time of $O(n \cdot 2^n)$ to build the cache. Instead, the following results show that this number is much smaller in many practical cases.

Lemma 1 *Let X_i be a node of $\mathcal{G}'$, a DAG for a BN where $PA_i = J'$. Suppose $J \subset J'$ is such that $s_i(J) > s_i(J')$. Then J' is not the parent set of X_i in the optimal DAG.*

Proof. Take a graph $\mathcal{G}$ that differs from $\mathcal{G}'$ only on $PA_i = J$, which is also a DAG (as the removal of some arcs does not create cycles) and $s(\mathcal{G}) = \sum_{j \neq i} s_j(PA_j) + s_i(J) > \sum_{j \neq i} s_j(PA_j) + s_i(J') = s(\mathcal{G}')$. Hence any DAG $\mathcal{G}'$ such that $PA_i = J'$ has a subgraph $\mathcal{G}$ with a better score than $\mathcal{G}'$, and thus J' is not the optimal parent configuration for X_i. □

Lemma 1 is quite simple but very useful to discard elements from the cache of X_i. However, it does not tell anything about supersets of J', that is, we still need to compute all the possible parent configurations and later verify which of them can be removed. Next theorems handle this issue.

Theorem 1 *Using MDL or AIC as score function and assuming $N \geq 4$, take $\mathcal{G}$ and $\mathcal{G}'$ DAGs such that $\mathcal{G}$ is a subgraph of $\mathcal{G}'$. If $\mathcal{G}$ is such that $\prod_{j \in PA_i} r_j \geq N$, for some X_i, and X_i has a proper superset of parents in $\mathcal{G}'$ w.r.t. $\mathcal{G}$, then $\mathcal{G}'$ is not an optimal structure.*

Proof.[1] Take a DAG $\mathcal{G}$ such that $J = PA_i$ for a node X_i, and take $\mathcal{G}'$ equal to $\mathcal{G}$ except that it contains an extra node in $J^{new} = PA_i$, that is, in $\mathcal{G}'$ we have $J^{new} = J \cup \{X_e\}$. Note that the difference in the scores of the two graphs are restricted to $s_i(\cdot)$. In the graph $\mathcal{G}'$, $L_i(J^{new})$ will certainly not decrease and $t_i(J^{new})$ will increase, both with respect to the values for $\mathcal{G}$. The difference in the scores will be $s_i(J^{new}) - s_i(J)$, which equals to

$$L_i(J^{new}) - t_i(J^{new}) - (L_i(J) - t_i(J)) \leq$$

$$-\sum_{j=1}^{q_i} \sum_{i=1}^{r_i} n_{ijk} \log \theta_{ijk} - t_i(J^{new}) + t_i(J) \leq$$

$$\sum_{j=1}^{q_i} n_{ij} \left(-\sum_{i=1}^{r_i} \frac{n_{ijk}}{n_{ij}} \log \frac{n_{ijk}}{n_{ij}} \right) - t_i(J^{new}) + t_i(J) \leq$$

$$\sum_{j=1}^{q_i} n_{ij} H(\theta_{ij}) - t_i(J^{new}) + t_i(J) \leq$$

$$\sum_{j=1}^{q_i} n_{ij} \log r_i - q_i \cdot (r_e - 1) \cdot (r_i - 1) \cdot W$$

The first step uses the fact that $L_i(J^{new})$ is negative, the second step uses that fact that $\hat{\theta}_{ijk} = \frac{n_{ijk}}{n_{ij}}$, with $n_{ij} = \sum_{i=1}^{r_i} n_{ijk}$, is the value that maximizes $L_i(\cdot)$, and the last step uses the fact that the entropy of a discrete distribution is less than the log of its number of categories. Finally, $\mathcal{G}$ is a better graph than $\mathcal{G}'$ if the last equation is negative, which happens if $q_i \cdot (r_e - 1) \cdot (r_i - 1) \cdot W \geq N \log r_i$. Because $r_i \geq 2 \Rightarrow r_i - 1 \geq \log r_i$, and $N \geq 4 \Rightarrow \frac{\log N}{2} \geq 1$ (the W of the MDL case), we have that $q_i = \prod_{j \in J} r_j \geq N$ ensures that $s_i(J^{new}) < s_i(J)$, which implies that the graph $\mathcal{G}'$ cannot be optimal. □

Corollary 1 *In the optimal structure $\mathcal{G}$, each node has at most $O(\log N)$ parents.*

Proof. It follows directly from Theorem 1 and the fact that $r_i \geq 2$, for all X_i. □

Theorem 1 and Corollary 1 ensures that the cache stores at most $O(\binom{n-1}{\log N})$ elements for each variable (all combinations up to $\log N$ parents). Although it does not help us to improve the theoretical size bound, Lemma 2 gives us even less elements.

[1]Another similar proof appears in (Bouckaert, 1994), but it leads directly to the conclusion of Corollary 1. The intermediate result is algorithmically important.

Lemma 2 *Let X_i be a node with $J \subset J'$ two possible parent sets such that $t_i(J') + s_i(J) > 0$. Then J' and all supersets $J'' \supset J'$ are not optimal parent configurations for X_i.*

Proof. Because $L_i(\cdot)$ is a negative function, $t_i(J') + s_i(J) > 0 \Rightarrow -t_i(J') - s_i(J) < 0 \Rightarrow (L_i(J') - t_i(J')) - s_i(J) < 0 \Rightarrow s_i(J') < s_i(J)$. Using Lemma 1, we have that J' is not the optimal parent set for X_i. The result also follows for any $J'' \supset J$, as we know that $t_i(J'') > t_i(J')$. □

Thus, the idea is to check the validity of Lemma 2 every time the score of a parent set J' of X_i is about to be computed, discarding J' and all supersets whenever possible. This result allows us to stop computing scores for J' and all its supersets. Lemma 1 is stronger, but regards a comparison between exactly two parent configuration. Nevertheless, Lemma 1 can be applied to the final cache to remove all certainly useless parent configurations. As we see in Section 5, the practical size of the cache after these properties is small even for large networks. Lemma 1 is also valid for other decomposable functions, including BD and derivations (e.g. BDe, BDeu), so the benefits shall apply to those scores too, and the memory requirements will be reduced. The other theorems need assumptions about the initial N and the choice of priors. Further discussion is left for future work because of lack of space.

3. Constraints

An additional way to reduce the space of possible DAGs is to consider some constraints provided by experts. We work with two main types of constraints: constraints on parameters that define rules about the probability values inside the local distributions of the network, and structural constraints that specify where arcs may or may not be included.

3.1. Parameter Constraints

We work with a general definition of parameter constraint, where any convex constraint is allowed. If θ_{i,PA_i} is the parameter vector of the node X_i with parent set PA_i, then a *convex constraint* is defined as $h(\theta_{i,PA_i}) \leq 0$, where $h : \Omega_{\theta_{i,PA_i}} \to \mathcal{R}$ is a convex function over θ_{i,PA_i}. This definition includes many well known constraints, for example from Qualitative Probabilistic Networks (QPN) (Wellman, 1990): *qualitative influences* define some knowledge about the state of a variable given the state of another, which roughly means that observing a greater state for a parent X_a of a variable X_b makes more likely to have greater states in X_b (for any parent configuration except for X_a). For example, $\theta_{bj_2 2} \geq \theta_{bj_1 2}$, where $j_k \doteq \{x_a^k, pa_b^{j_*}\}$ and j_* is an index ranging over all parent configurations except for X_a. In this case, observing x_a^2 makes more likely to have x_b^2. A *negative influence* is obtained by replacing the inequality operator $\geq$ by $\leq$, and a *zero influence* is obtained by changing inequality to an equality. Other constraints such as *synergies* (Wellman, 1990) are also linear and local to a single node.

Although we allow the parameter constraints that are general, we have the following restriction about them: if a constraint is specified for a node X_i and a set of parents J, then the actual parent set PA_i has to be a superset of J. Furthermore, we have a peculiar interpretation for each constraint C as follows: if $J \subset PA_i$ (proper subset), then the parameter constraint must hold for all configurations of the parents of X_i that do not belong to J. For example, suppose X_1 has X_2 and X_3 as parents (all of them binary), and the following constraint h was defined on X_1: $p(x_1^2|x_2^2x_3^2) + 2 \cdot p(x_1^2|x_2^2x_3^1) \leq 1$. If a new node X_4 is included as parent of X_1, the constraint h becomes the two following constraints:

$$\begin{aligned} p(x_1^2|x_2^2x_3^2x_4^1) + 2 \cdot p(x_1^2|x_2^2x_3^1x_4^1) &\leq 1, \\ p(x_1^2|x_2^2x_3^2x_4^2) + 2 \cdot p(x_1^2|x_2^2x_3^1x_4^2) &\leq 1, \end{aligned}$$

that is, h holds for each state of X_4. For example if another parent X_5 is included, then four constraints would be enforced with all possible combinations. This interpretation for constraints is in line with the definition of qualitative constraints of QPNs, and most importantly, it allows us to treat the constraints in a principled way for each set of parents. It means that the constraint must hold for all configurations of parents not involved in the constraint, which can be also interpreted as *other parents are not relevant and the constraint is valid for each one of their configurations.*

3.2. Structural constraints

Besides probabilistic constraints, we work with structural constraints on the possible graphs. These constraints help to reduce the search space and are available in many situations. We work with the following rules:

- $indegree(X_j, k, op)$, where $op \in \{\text{lt}, \text{eq}\}$ and k an integer, means that the node X_j must have *less than* (when $op = \text{lt}$) or *equal to* (when $op = \text{eq}$) k parents.
- $arc(X_i, X_j)$ indicates that the node X_i must be a parent of X_j.
- Operators *or* ($\vee$) and *not* ($\neg$) are used to form the rules. The *and* operator is not explicitly used as we assume that each constraint is in disjunctive normal form.

For example, the constraints $\forall_{i \neq c, j \neq c}\ \neg arc(X_i, X_j)$ and $indegree(X_c, 0, \text{eq})$ impose that only arcs from node X_c to the others are possible, and that X_c is a root node, that is, a Naive Bayes structure will be learned. The procedure will also act as a feature selection procedure by letting some variables unlinked. Note that the symbol $\forall$ just employed is not part of the language but is used for easy of expose (in fact it is necessary to write down every constraint defined by such construction). As another example, the constraints $\forall_{j \neq c}\ indegree(X_j, 3, \text{lt})$, $indegree(X_c, 0, \text{eq})$, and $\forall_{j \neq c}\ indegree(X_j, 0, \text{eq}) \vee arc(X_c, X_j)$ ensure that all nodes have X_c as parent, or no parent at all. Besides X_c, each node may have at most one other parent, and X_c is a root node. This learns the structure of a Tree-augmented Naive (TAN) classifier, also performing a kind of feature selection (some variables may end up unlinked). In fact, it learns a forest of trees, as we have not imposed that all variables must be linked.

3.3. Dealing with constraints

All constraints in previous examples can be imposed during the construction of the cache, because they involve just a single node each. In essence, parent sets of a node X_i that do violate some constraint are not stored in the cache, and this can be checked during the cache construction. On the other hand, constraints such as $arc(X_1, X_2) \vee arc(X_2, X_3)$ cannot be imposed in that stage, as they impose a non-local condition (the arcs go to distinct variables, namely X_2 and X_3), because the cache construction is essentially a local procedure with respect to each variable. Such constraints that involve distinct nodes can be verified during the B&B phase, so they are addressed later.

Regarding parameter constraints, we compute the scores using a constrained optimization problem, i.e. maximize the score function subject to simplex equality constraints and all parameter constraints defined by the user.

$$\begin{aligned} &\max_{\theta_i} L_i(\theta_i) - t_i(PA_i) \\ &\qquad \text{subject to} \quad \forall_{j=1\ldots q_i} \quad g_{ij}(\theta_{ij}) = 0, \\ &\qquad\qquad\qquad\quad \forall_{z=1\ldots m_{h_i}} \quad h_{iz}(\theta_i) \leq 0, \end{aligned} \tag{2}$$

where $g_{ij}(\theta_{ij}) = -1 + \sum_{k=1}^{r_i} \theta_{ijk}$ imposes that distributions defined for each variable given a parent configuration sum one over all variable states, and the m_{hi} convex constraints h_{iz} define the space of feasible parameters for the node X_i. This is possible because: (1) we have assumed that a constraint over $p(x_i^k|x_{i_1}^{k_1},\ldots,x_{i_t}^{k_t})$ forces $X_{i_1},\ldots,X_{i_t} \subseteq PA_i$, that is, when a parameter constraint is imposed, the parent set of the node must contain at least the variables involved in the constraint; (2) the optimization is computed for every possible parent set, that is, PA_i is known in the moment to write down the optimization problem, which is solved for each X_i and each set PA_i. We use the optimization package of (Birgin et al., 2000).

Theorem 2 *Using MDL or AIC as score function and assuming $N \geq 4$, take $\mathcal{G}$ and $\mathcal{G}'$ as DAGs such that $\mathcal{G}$ is a subgraph of $\mathcal{G}'$. Suppose that both $\mathcal{G}$ and $\mathcal{G}'$ respect the same set of parameter and structural constraints. If $\mathcal{G}$ is such that $\prod_{j\in PA_i} r_j \geq N$, for some X_i, and X_i has a proper superset of parents in $\mathcal{G}'$ w.r.t. $\mathcal{G}$, then $\mathcal{G}'$ is not an optimal structure.*

Proof. Just note that all derivations in Theorem 1 are also valid in the case of constraints. The only difference that deserves a comment is $\hat{\theta}_{ijk} = \frac{n_{ijk}}{n_{ij}}$, which may be an unfeasible point for the optimization (2), because the latter contains parameter constraints that might reduce the parameter space (besides the normal constraints of the maximum log-likelihood problem). As $\hat{\theta}_{ijk}$ is just used as an upper value for the log-likelihood function, and the constrained version can just obtain smaller objective values than the unconstrained version, $\frac{n_{ijk}}{n_{ij}}$ is an upper bound also for the constrained case. Thus, the derivation of Theorem 1 is valid even with constraints. □

Corollary 1 and Lemmas 1 and 2 are also valid in this setting. The proof of Corollary 1 is straightforward, as it only depends on Theorem 1, while for Lemmas 1 and 2 we need just to ensure that all the parent configurations that are discussed there respect the constraints.

4. Constrained B&B algorithm

In this section we describe the B&B algorithm used to find the best structure of the BN and comment on its complexity, correctness, and some extensions and particular cases. The notation (and initialization of the algorithm) is as follows: $C : (X_i, PA_i) \to \mathcal{R}$ is the cache with the scores for all the variables and their possible parent configurations (using Theorem 1 and Lemmas 1 and 2 to have a reduced size); $\mathcal{G}$ is the graph created taking the best parent configuration for each node without checking for acyclicity (so it is not necessarily a DAG), and s is the score of $\mathcal{G}$; $\mathcal{H}$ is an initially empty matrix containing, for each possible arc between nodes, a mark stating that the arc must be present, or is prohibited, or is free (may be present or not); Q is a priority queue of triples $(\mathcal{G},\mathcal{H},s)$, ordered by s (initially it contains a single triple with $\mathcal{G}$, $\mathcal{H}$ and s just mentioned; and finally $(\mathcal{G}_{best}, s_{best})$ is the best DAG and score found so far (s_{best} is initialized with $-\infty$). The main loop is as follows:

While Q is not empty, do

1. Remove the peek $(\mathcal{G}_{cur},\mathcal{H}_{cur},s_{cur})$ of Q. If $s \leq s_{best}$ (worst than an already known solution), then start the loop again. If $\mathcal{G}_{cur}$ is a DAG and satisfies all structural constraints, update $(\mathcal{G}_{best}, s_{best})$ with $(\mathcal{G}_{cur}, s_{cur})$ and start the loop again.
2. Take $v = (X_{a_1} \to X_{a_2} \to \ldots \to X_{a_{q+1}})$, with $a_1 = a_{q+1}$, is a directed cycle of $\mathcal{G}_{cur}$.
3. For $y = 1,\ldots,q$, do
   - Mark on $\mathcal{H}_{cur}$ that the arc $X_{a_y} \to X_{a_{y+1}}$ is prohibited.
   - Recompute $(\mathcal{G}, s)$ from $(\mathcal{G}_{cur}, s_{cur})$ such that the parents of $X_{a_{y+1}}$ in $\mathcal{G}$ comply with this restriction and with $\mathcal{H}_{cur}$. Furthermore, the subgraph of $\mathcal{G}$ formed by arcs that are demanded by $\mathcal{H}_{cur}$ (those that have a mark *must exist*) must comply with the structural constraints (it might be impossible to get such graph. In such case, go to the last bullet). Use the values in the cache $C(X_{a_{y+1}}, PA_{a_{y+1}})$ to avoid recomputing scores.
   - Include the triple $(\mathcal{G},\mathcal{H}_{cur},s)$ into Q.
   - Mark on $\mathcal{H}_{cur}$ that the arc $X_{a_y} \to X_{a_{y+1}}$ must be present and that the sibling arc $X_{a_{y+1}} \to X_{a_y}$ is prohibited, and continue.

The algorithm uses a B&B search where each case to be solved is a relaxation of a DAG, that is, they may contain cycles. At each step, a graph is picked up from a priority queue, and it is verified if it is a DAG. In such case, it is a feasible structure for the network and we compare its score against the best score so far (which is updated if needed). Otherwise, there must be a directed cycle in the graph, which is then broken into subcases by forcing some arcs to be absent/present. Each subcase is put in the queue to be processed. The procedure stops when the queue is empty. Note that every time we break a cycle, the subcases that are created are independent, that is, the sets of graphs that respect $\mathcal{H}$ for each subcase are disjoint. We obtain this fact by properly breaking the cycles: when $v = (X_{a_1} \to X_{a_2} \to \ldots \to X_{a_{q+1}})$ is detected, we create q

subcases such that the first does not contain $X_{a_1} \to X_{a_2}$ (but may contain the other arcs of that cycle), the second case certainly contains $X_{a_1} \to X_{a_2}$, but $X_{a_2} \to X_{a_3}$ is prohibited (so they are disjoint because of the difference in the presence of the first arc), and so on such that the y-th case certainly contains $X_{a_{y'}} \to X_{a_{y'+1}}$ for all $y' < y$ and prohibits $X_{a_y} \to X_{a_{y+1}}$. This idea ensures that we never process the same graph twice. So the algorithm runs at most $\prod_i |C(X_i)|$ steps, where $|C(X_i)|$ is the size of the cache for X_i.

B&B can be stopped at any time and the current best solution as well as an upper bound for the global best score are available. This stopping criterion might be based on the number of steps, time and/or memory consumption. Moreover, the algorithm can be easily parallelized. We can split the content of the priority queue into many different tasks. No shared memory needs to exist among tasks if each one has its own version of the cache. The only data structure that needs consideration is the queue, which from time to time must be balanced between tasks. With a message-passing idea that avoids using process locks, the gain of parallelization is linear in the number of tasks. As far as we know, best known exact methods are not easily parallelized, they do not deal with constraints, and they do not provide lower and upper estimates of the best structure if stopped early. If run until it ends, the proposed method gives a global optimum solution for the structure learning problem.

Some particular cases of the algorithm are worth mentioning. If we fix an ordering for the variables such that all the arcs must link a node towards another non-precedent in the ordering (this is a common idea in many approximate methods), the proposed algorithm does not perform any branch, as the ordering implies acyclicity, and so the initial solution is already the best. The performance would be proportional to the time to create the cache. On the other hand, bounding the maximum number of parents of a node is relevant only for hardest inputs, as it would imply a bound on the cache size, which is already empirically small.

5. Experiments

We perform experiments to show the benefits of the reduced cache and search space and the gains of constraints.[2] First, we use data sets available at the UCI repository (Asuncion & Newman, 2007). Lines with missing data are removed and continuous variables are discretized over the mean into binary variables. The data sets are: *adult* (15 variables and 30162 instances), *car* (7 variables and 1728 instances) letter (17 variables and 20000 instances), lung (57 variables and 27 instances), mushroom (23 variables and 1868 instances), nursery (9 variables and 12960 instances), Wisconsin Diagnostic Breast Cancer or *wdbc* (31 variables and 569 instances), zoo (17 variables and 101 instances). No constraints are employed in this phase as we intend to show the benefits of the properties earlier discussed.

Table 1 presents the cache construction results, applying Theorem 1 and Lemmas 1 and 2. Its columns show the data set name, the number of steps the procedure spends to build the cache (a step equals to a call to the score function for a single variable and a parent configuration), the time in seconds, the size of the generated cache (number of scores stored, the memory consumption is actually $O(n)$ times that number), and finally the size of the cache if all scores were computed. Note that the reduction is huge. Although in the next we are going to discuss three distinct algorithms, the benefits of the application of these results imply in performance gain for other algorithms in the literature to learn BN structures. It is also possible to analyze the search space reduction implied by these results by looking columns 2 and 3 of Table 2.

Table 1. Cache sizes (number of stored scores) and time (in seconds) to build them for many networks and data sizes. Steps represent the number of local (single node given a parent set) score evaluations.

name	steps	time(s)	size	$n2^n$
adult	30058	182.09	672	$2^{17.9}$
car	335	0.09	24	$2^{8.8}$
letter	534230	2321.46	41562	$2^{20.1}$
lung	43592	1.33	3753	$2^{61.8}$
mushroom	140694	72.13	8217	$2^{26.5}$
nursery	1905	3.94	49	$2^{11.2}$
wdbc	1692158	351.04	7482	2^{35}
zoo	9118	0.31	1875	$2^{20.1}$

In Table 2, we show results of three distinct algorithms: the B&B described in Section 4, the dynamic programming (DP) idea of (Silander & Myllymaki, 2006), and an algorithm that picks variable orderings randomly and then find the best structure such that all arcs link a node towards another that is not precedent in the ordering. This last algorithm (named OS) is similar to K2 algorithm with random orderings, but it is always better because a global optimum is found for each ordering.[3] The scores obtained by each algorithm (in percentage against the value obtained by B&B) and

[2]The software is available online through the web address *http://www.ecse.rpi.edu/∼cvrl/structlearning.html*

[3]We have run a hill-climbing approach (which is also benefited by ideas presented in this paper), but its accuracy was worse than OS. We omit it because of lack of space.

Table 2. Comparison of MDL scores among B&B, dynamic programming (DP), and ordering sampling (one thousand times). *Fail* means that it could not solve the problem within 10 million steps. DP and OS scores are in percentage w.r.t. the score of B&B (positive percentage means worse than B&B and negative percentage means better).

	search	reduced	B&B			DP		OS	
network	space	space	score	gap	time(s)	score	time(s)	score	time(s)
adult	2^{210}	2^{71}	-286902.8	5.5%	150.3	0.0%	0.77	0.1%	0.17
car	2^{42}	2^{10}	-13100.5	0.0%	0.01	0.0%	0.01	0.0%	0.01
letter	2^{272}	2^{188}	-173716.2	8.1%	574.1	-0.6%	22.8	1.0%	0.75
lung	2^{3192}	2^{330}	-1146.9	2.5%	907.1	*Fail*	*Fail*	1.0%	0.13
mushroom	2^{506}	2^{180}	-12834.9	15.3%	239.8	*Fail*	*Fail*	1.0%	0.12
nursery	2^{72}	2^{17}	-126283.2	0.0%	0.04	0.0%	0.04	0.0%	0.04
wdbc	2^{930}	2^{216}	-3053.1	13.6%	333.5	*Fail*	*Fail*	0.8%	0.13
zoo	2^{272}	2^{111}	-773.4	0.0%	5.2	0.0%	3.5	1.0%	0.03

the corresponding spent time are presented (excluding the cache construction). A limit of ten million steps is given to each method (steps here are considered as the number of queries to the cache). It is also presented the reduced space where B&B performs its search, as well as the maximum gap of the solution. This gap is obtained by the relaxed version of the problem. So we can guarantee that the global optimal solution is within this gap (even though the solution found by the B&B may already be the best, as shown in the first line of the table). With the reduced cache presented here, finding the best structure for a given ordering is very fast, so it is possible to run OS over millions of orderings in a short period of time.

Table 3. B&B procedure learning TANs. Time (in seconds) to find the global optimum, cache size (number of stored scores) and (reduced) space for B&B search.

network	time(s)	cache size	space
adult	0.26	114	2^{39}
car	0.01	14	$2^{6.2}$
letter	0.32	233	2^{61}
lung	0.26	136	2^{51}
mushroom	0.71	398	2^{88}
nursery	0.06	26	2^{12}
wdbc	361.64	361	2^{99}

Some additional comments are worth. DP can solve the *mushroom* set in less than 10 minutes if we drop the limit of steps. The expectation for *wdbc* is around four days. Hence, we cannot expect to obtain an answer in larger cases, such as *lung*. It is clear that, in worst case, the number of steps of DP is smaller than that of B&B. Nevertheless, B&B eventually bounds some regions without processing them, provides an upper bound at each iteration, and does not suffer from memory exhaustion as DP. This makes the method applicable even to very large settings. Still, DP seems a good choice for small n. When n is large (more than 35), DP will not finish in reasonable time, and hence will not provide any solution, while B&B still gives an approximation and a bound to the global optimum. About OS, if we sample one million times instead of one thousand as done before, its results improve and the global optimum is found also for *adult* and *mushroom* sets. Still, OS provides no guarantee or estimation about how far is the global optimum (here we know it has achieved the optimum because of the exact methods). It is worth noting that both DP and OS are benefited by the smaller cache.

Table 3 shows the results when we employ constraints to force the final network to be a Tree-augmented Naive Bayes (*zoo* was run, but it is not included because the unconstrained learned network was already TAN). Here the class is isolated in the data set and constraints are included as described in Section 3.2. Note that the cache size, the search space and consequently the time to solve the problems have all decreased. Finally, Table 4 has results for random data sets with predefined number of nodes and instances. A randomly created BN with at most $3n$ arcs is used to sample the data. Because of that, we are able to generate random parameter and structural constraints that are certainly valid for this *true* BN (approximately $n/2$ constraints for each case). The table contains the total time to run the problem and the size of the cache, together with the percentage of gain when using constraints. Note that the code was run in parallel with a number of tasks equals to n, otherwise an increase by a factor of n must be applied to the results in the table. We can see that the gain is recurrent in all cases (the constrained version has also less gap in all cases, although such number is not shown).

6. Conclusions

This paper describes a novel algorithm for learning BN structure from data and expert's knowledge. It integrates structural and parameter constraints with

Table 4. Results on random data sets generated from random networks. Time to solve (10 million steps) and size of the cache are presented for the *normal* unconstrained case and the percentage of gain when using constraints.

nodes(n)/	unconstrained			constrained gain	
instances	gap	time(s)	cache	time	cache
30/100	0%	0.06	125	67%	11.6%
30/500	0%	2.7	143	47.5%	26.5%
50/100	0%	0.26	310	31.4%	16.1%
50/500	0%	20.66	231	57.2%	29.8%
70/100	0%	4.58	1205	36.9%	18.8%
70/500	1.1%	356.9	666	38.4%	21.9%
100/100	0.5%	9.05	2201	47.5%	23.5%
100/500	1.4%	1370.4	726	50.2%	33.0%

data through a B&B procedure that guarantees global optimality with respect a decomposable score function. It is an any-time procedure in the sense that, if stopped early, it provides the best current solution found so far and a maximum error of such solution. The software is available as described in the experiments.

We also describe properties of the structure learning problem based on scoring DAGs that enable the B&B procedure presented here as well as other methods to work over a reduced search space and memory. Such properties allow the construction of a cache with all possible local scores of nodes and their parents without large memory consumption.

Because of the properties and the characteristics of the B&B method, even without constraints the B&B is more efficient than state-of-the-art exact methods for large domains. We show through experiments with randomly generated data and public data sets that problems with up to 70 nodes can be exactly processed in reasonable time, and problems with 100 nodes are handled within a small worst case error. These results surpass by far current methods, and may also help to improve other approximate methods and may have interesting practical applications, which we will pursue in future work.

Acknowledgments

This work is supported in part by the grant W911NF-06-1-0331 from the U.S. Army Research Office. The first author thanks also the project *Ticino in Rete*.

References

Asuncion, A., & Newman, D. (2007). UCI machine learning repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Birgin, E. G., Martínez, J. M., & Raydan, M. (2000). Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. on Optimiz.*, *10*, 1196–1211.

Bouckaert, R. (1994). Properties of bayesian belief network learning algorithms. *Conf. on Uncertainty in Artificial Intelligence* (pp. 102–109). M. Kaufmann.

Chickering, D., Meek, C., & Heckerman, D. (2003). Large-sample learning of bayesian networks is np-hard. *Conf. on Uncertainty in Artificial Intelligence* (pp. 124–13). M. Kaufmann.

Chickering, D. M. (2002). Optimal structure identification with greedy search. *J. of Machine Learning Research*, *3*, 507–554.

Cooper, G., & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Mach. Learning*, *9*, 309–347.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning bayesian networks: The combination of knowledge and statistical data. *Mach. Learning*, *20*, 197–243.

Koivisto, M. (2006). Advances in exact bayesian structure discovery in bayesian networks. *Conf. on Uncertainty in Artificial Intelligence* (pp. 241–248) AUAI Press.

Koivisto, M., Sood, K., & Chickering, M. (2004). Exact bayesian structure discovery in bayesian networks. *J. of Machine Learning Research*, *5*, 2004.

Silander, T., & Myllymaki, P. (2006). A simple approach for finding the globally optimal bayesian network structure. *Conf. on Uncertainty in Artificial Intelligence.* (pp. 445–452) AUAI Press.

Singh, A. P., & Moore, A. W. (2005). *Finding optimal bayesian networks by dynamic programming* (Technical Report). Carnegie Mellon Univ. CALD-05-106.

Suzuki, J. (1996). Learning bayesian belief networks based on the minimum description length principle: An efficient algorithm using the B&B technique. *Int. Conf. on Machine Learning* (pp. 462–470).

Teyssier, M., & Koller, D. (2005). Ordering-based search: A simple and effective algorithm for learning bayesian networks. *Conf. on Uncertainty in Artificial Intelligence.* (pp. 584–590) AUAI Press.

Tsamardinos, I., Brown, L. E., & Aliferis, C. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Mach. Learning*, *65*, 31–78.

Wellman, M. P. (1990). Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, *44*, 257–303.

Learning Limited Memory Influence Diagrams

Cassio P. de Campos and Qiang Ji

September 1, 2009

1 Introduction

An influence diagram is a graphical model for decision making under uncertainty [7]. It is composed by a directed graph where utility nodes are associated to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represent actions to be taken. Given an influence diagram, a strategy defines which decision to take at each node, given the information available at that moment. Each strategy has a corresponding expected utility. One important problem in influence diagrams is learning the model, which includes the elicitation of probability distributions for the chance nodes and utility functions for the utility nodes. The direct elicitation of such values using expert knowledge may be replaced by an automatic learning procedure when a data set of past events is available.

In this paper, we propose new ideas to learn the parameters of a *Limited Memory Influence Diagram*, or simply LIMID. LIMIDs represent a very general class of influence diagrams, and include the most traditional case (introduced initially by [7]) as subcase [8]. *Limited Memory* means that the assumption of *no-forgetting* usually employed in standard Influence Diagrams (that is, values of observed variables and decisions that have been taken are remembered at all later times) is relaxed. Because LIMIDs are general and do not have assumptions about no-forgetting and ordering for decisions, it is possible to efficiently convert diagrams that have such assumptions into LIMIDs. Hence, LIMIDs are more powerful (in the sense of expressiveness) than other influence diagrams. The benefits of having a more expressive model come with the additional computational cost. Thus, specialized algorithms that exploit LIMID's characteristics are needed.

We describe a learning procedure for LIMIDs that is motivated by previous work on the traditional influence diagram [10, 9, 1]. We assume that the decision maker takes rational decisions, which leads to the maximization of the expected utility. This model requires two types of information: the probabilities and the utilities of all possible outcomes of the decision problem. Probability values are obtained by standard learning procedures, usually borrowed from the theory of Bayesian networks.

The estimation of the utility function can be addressed by elicitation from experts, which need to answer a sequence of questions about their preferences, or by automatic procedures that receive as input the past behavior of experts through a data set of cases. Dealing with the experts may be difficult and generate errors, as previously reported [6]. On the other hand, the data set of past cases cam be used to create a set of constraints on the space of possible utility functions, from where later a conservative (e.g. by using a maximin approach) or a more aggressive (e.g. by taking any admissible function inside the set) utility function is drawn. In the literature, we find heuristics [9] and probabilistic approaches [1]. Here we assume that a database of past events, decisions and rewards is available to train the influence diagram, which is a reasonable assumption in military problems, and we use this database to learn both the probability values and the utility functions. The behavior of the experts, encoded in the database, is employed to create constraints on the utility function of the problem. Such constraints are integrated in the strategy selection process using the ideas we have developed [5].

2 LIMIDs: Limited Memory Influence Diagrams

A Limited Memory Influence Diagram $\mathcal{I}$ is composed by a directed acyclic graph $(\mathcal{V}, E)$ where nodes are partitioned in three types: chance, decision and utility nodes. Let $\mathcal{C}$, $\mathcal{D}$ and $\mathcal{U}$ be the set of chance, decision and utility nodes, respectively, and let $\mathcal{X} = \mathcal{C} \cup \mathcal{D}$. Links of E characterize dependencies among nodes. Explicitly, links toward a chance node indicate probabilistic dependence of the node on its parents; links toward a decision node indicate which information is available to take such decision, and links toward utility nodes represent that an utility for those parents is to be considered (utility nodes may not have children). Associated to each node, there are some parameters:

1. A *chance node* has an associated categorical random variable C with finite domain Ω_C and conditional probability distributions $p(C|\pi_j(C))$, for each configuration $\pi_j(C)$ of its parents $\pi(C)$ in the graph. j is used to indicate a configuration of the parents of C, that is, $\pi_j(C) \in \Omega_{\pi(C)}$, where the notation $\Omega_{\mathcal{V}'} = \times_{V \in \mathcal{V}'} \Omega_V$, for any $\mathcal{V}' \subseteq \mathcal{V}$.

2. A *decision node* D is associated to a finite set of mutually exclusive alternatives Ω_D. Parents of D describe the information that is available at the moment on which decision D has to be taken.

3. An *utility node* U is associated to a rational function $f_U : \Omega_{\pi(U)} \to \mathcal{Q}$. The value corresponding to a parent configuration is the profit (cost is viewed as negative profit) of such parent configuration. Utility nodes have no children.

A simple example is depicted in Figure 1. Decision nodes are represented by rectangles, chance nodes by ellipses and utility nodes by diamonds. *do_ground_attack* has an associated cost, which is depicted by the corresponding utility node. The same is modeled for *bomb_bridge*. The goal is to achieve *territory_occupation*, which also has an utility (the profit of the goal). *ground_attack* and *bridge_condition*

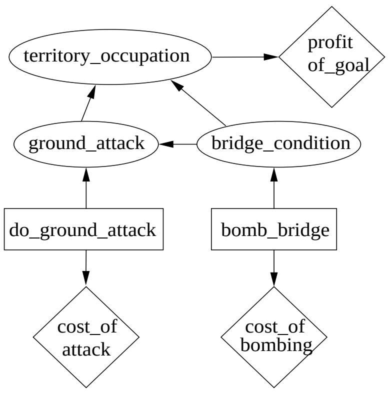


Figure 1: Simple Influence Diagram example.

represent the uncertain outcomes of the corresponding actions. To simplify notation, we denote the nodes by their initial letters as follows: *do_ground_attack* is denoted by DGA, *bomb_bridge* by BB, *territory_occupation* by TO, *ground_attack* by GA, and so on.

Note that it is not assumed to exist a known ordering (among decision nodes) on which decisions must be taken, as it is done in simpler versions of influence diagrams. Although decision nodes have no parents in the example of Figure 1, this is not a restriction of the model.

A *policy* δ_D for the decision node D is a function $\delta_D : \Omega_{D \cup \pi(D)} \to [0,1]$ defined for each alternative of D and each configuration of $\pi(D)$ such that, for each $\pi_j(D) \in \Omega_{\pi(D)}$ we have $\sum_{d \in \Omega_D} \delta_D(d, \pi_j(D)) = 1$. A *pure policy* is a policy such that its image is integer ($\delta_D : \Omega_{D \cup \pi(D)} \to \{0,1\}$), and thus specifies with certainty which action (alternative of D) is taken for each parent configuration (in a pure policy, only one $\delta_D(d, \pi_j(D))$ for each $\pi_j(D)$ will be non-zero as they sum 1). A *strategy* Δ is a set of policies $\{\delta_D : D \in \mathcal{D}\}$, one for each decision node of the diagram. A *pure strategy* is composed only by pure policies.

The expected utility $EU(\Delta)$ of a strategy Δ is evaluated through the following equation:

$$\sum_{\mathbf{x} \in \Omega_{\mathcal{X}}} \left(\prod_C p(x_C | \pi_j(C)) \prod_D \delta_D(x_D) \sum_U f_U(\pi_{j'}(U)) \right), \tag{1}$$

where x_C, $\pi_j(C)$, x_D and $\pi_{j'}(U)$ are respectively the projections of $\mathbf{x}$ in Ω_C, $\Omega_{\pi(C)}$, $\Omega_{D \cup \pi(D)}$ and $\Omega_{\pi(U)}$. This equation means that, given a strategy, its expected utility is the sum of the utility values weighted by the probability of each diagram configuration (for all configurations). The maximum expected utility is obtained over all possible strategies:

$$MEU = \max_{\Delta} EU(\Delta).$$

Chances			Decisions	
TO	*GA*	*BC*	*DGA*	*BB*
1	1	1	1	0
1	1	1	0	1
0	1	1	0	1
0	0	1	1	1
...	...	...	...	...

Table 1: Example of database for LIMID of Figure 1 where outcomes of the chance and decision nodes are available.

Optimal strategy		Utilities		
DGA	*BB*	*COA*	*COB*	*POG*
1	1	-100	-50	200
?	1	0	-60	180
1	0	-80	0	-30
0	0	0	0	-50
...	...	...	...	...

Table 2: Example of database for LIMID of Figure 1 where local utility rewards are available when the optimal strategy is used.

The problem of *strategy selection* is to obtain the strategy that maximizes its expected utility, that is, argmax $\max_\Delta EU(\Delta)$.

Algorithms to solve the strategy selection problem assume that all parameters of chance nodes and utility nodes are known and fixed. However, eliciting all such parameters from experts is a hard task, and usually can be done in small sized LIMIDs. Next section address this problem, providing an approach to learn parameters from past experience.

3 Learning LIMIDs

As discussed, we use a database containing past experience to learn the parameters of the LIMID. The data are divided into two parts: $DB_\mathcal{C}$ is a set of samples of the chance and decision nodes (note that in these data, the decisions do not necessarily represent optimal decisions); $DB_\mathcal{D}$ is a set of samples containing an optimal strategy (which is defined by a set of optimal decisions) and the utility values obtained at each utility node when the optimal strategy is employed. Using the LIMID of Figure 1, examples of data that would be available to learn its parameters are presented in Tables 1 and 2 (the nodes of Figure 1 are denoted by their initial letters, as explained before). True is denoted by a one, and false is indicated by a zero.

Using the database of chance and decision variables, standard techniques can be employed to find the most probable values for the parameters of the chance nodes, denoted $\mathbb{P} = \{p(C|\pi_j(C))\}_{\forall C}$. One way to quantify the result is by the log likelihood function $\log(p(DB_{\mathcal{C}}|\mathbb{P}))$. Because utility nodes have no children, chance nodes can only have other chance nodes and/or decision nodes as parents. Assuming that samples are drawn independently from the underlying distribution, we can use the decomposition property of the log likelihood function to maximize $\log \prod_{ijk} p(x_{ik}|\pi_{ij})^{n_{ijk}}$, where $x_{ik} \in \Omega_{C_i}$, n_{ijk} indicates how many elements of $DB_{\mathcal{C}}$ contain both x_{ik} and π_{ij}, where $\pi_{ij} \in \Omega_{\pi_{C_i}}$ is a joint configuration of the parents of C_i and can involve chance and decision nodes. Maximum likelihood estimation has its optimum at $p(x_{ik}|\pi_{ij}) = \frac{n_{ijk}}{\sum_k n_{ijk}}$. One may also use a Bayesian Dirichlet model instead of maximum likelihood estimation, just as it is done in Bayesian networks.

3.1 Learning utility functions

Our goal here is to estimate utility function $f_U : \Omega_{\pi(U)} \to \mathcal{Q}$ for each utility node U, based on past optimal decisions. Note that our database $DB_{\mathcal{D}}$ contains only optimal strategies and their utilities at that moment. If we assume that $DB_{\mathcal{D}}$ contains a wide range (in the sense of covering all the utility space) of decisions and corresponding utility values, an estimation based on expectation suffices. For instance, we take the average of the utility values that appear in $DB_{\mathcal{D}}$, for each configuration $\pi_j(U) \in \Omega_{\pi(U)}$, and construct a set of constraints that relate the utility values for distinct elements $\pi_j(U)$ as follows:

- For each sample $l = (l_{\mathcal{D}}, l_{\mathcal{U}})$ in $DB_{\mathcal{D}}$, let l_U (for each $U \in \mathcal{U}$) be the observed utility value of node U and $l_{\mathcal{D}}$ the decision values of sample l. Build the constraint:

$$\forall U : \quad E[l_U|l_{\mathcal{D}}] = \sum_{x \in \Omega \pi(U) \setminus \mathcal{D}} f_U(x \cup l_{\mathcal{D}}) \cdot p(x|l_{\mathcal{D}}), \tag{2}$$

  where $p(x|l_{\mathcal{D}})$ is calculated a priori given that all elements of $\mathbb{P}$ are already known, and $E[l_U|l_{\mathcal{D}}]$ accounts for the expectation of l_U over the samples l that are compatible with $l_{\mathcal{D}}$.

This way we have a set of linear constraints to define the utility function of each node U. Note that if a utility node has only decision nodes as parents (no chance node as parent), then Equation (2) simplifies to an equality without summation. For example, if we take the first line of Table 2, it implies the following constraints:

$$-90 = E[l_{\text{COA}}|\text{DGA=1}] = f_{\text{COA}}(\text{DGA=1}),$$
$$-55 = E[l_{\text{COB}}|\text{BB=1}] = f_{\text{COB}}(\text{BB=1}). \tag{3}$$

The situation becomes more difficult when the optimal strategy is only partially observed, that is, missing values may appear in the optimal decisions of a sample in $DB_{\mathcal{D}}$. In this case, we proceed in a similar way, but considering all possible completions of the data [11]. Such approach leads to the following constraints:

- For each sample $l = (l_{\mathcal{D}}, l_{\mathcal{U}})$ in $DB_{\mathcal{D}}$, let $l_{\mathcal{D}} = l^o_{\mathcal{D}} \cup l'_{\mathcal{D}}$, with $l^o_{\mathcal{D}}$ the observed part and $l'_{\mathcal{D}}$ the missing part. Build the constraint:

  $\forall U, l_{\mathcal{D}} :\ \underline{E}[l_U | l_{\mathcal{D}}] \leq$

  $$\sum_{x \in \Omega\pi(U) \setminus \mathcal{D}} \left(\sum_{l \in \Omega_{l'_{\mathcal{D}}}} I_{l'_{\mathcal{D}}}(l) \cdot f_U(x \cup l^o_{\mathcal{D}} \cup l) \cdot p(x | l^o_{\mathcal{D}} \cup l) \right) \leq \overline{E}[l_U | l_{\mathcal{D}}], \quad (4)$$

  where $I_{l'_{\mathcal{D}}}(\cdot)$ is an indicator function that is one only when l is the completion of l', and $\underline{E}[l_U | l_{\mathcal{D}}], \overline{E}[l_U | l_{\mathcal{D}}]$ are computed as the minimum and maximum values of the expectation of l_U considering all possible completions. We treat $I_{l'_{\mathcal{D}}}(\cdot)$ as additional boolean variables of the problem, which are going to be assigned later by the strategy selection procedure. Note that $\sum_l I_{l'_{\mathcal{D}}}(l) = 1$.

It is possible to use such idea with additional boolean variables because we resort to the procedures we have developed for strategy selection [5], where the problem is tackled by integer programming techniques (and thus the boolean variables are trivially included in the optimization problem). As an example, take the second line of Table 2 with respect to the node DGA. As it is missing, we have the following constraints:

$$-90 \leq \sum_{v \in \{0,1\}} I_{l_{\text{DGA}}}(v) \cdot f_{\text{COA}}(\text{DGA}{=}v) \leq -60,$$

$$0 \leq \sum_{v \in \{0,1\}} I_{l_{\text{DGA}}}(v) \cdot f_{\text{COA}}(\text{DGA}{=}v) \leq 0. \quad (5)$$

where $I_{l_{\text{DGA}}}$ is treat as a boolean variable, that is, depending on the value assigned to it, the data is completed in a different way. This completion is automatically conducted by the optimization of Equation (1) subject to Equations (2) and (4), so the same algorithm we have developed before [5] is used to select the best strategy over this now learned LIMID.

3.2 The optimization problem of the EBO example

To illustrate, we write down the optimization problem of the example in Figure 1. The utility functions are divided by the largest value in the database so they certainly belong to the interval $[0, 1]$. This division does not affect the choice of the optimal strategy [5]. Decision nodes are replaced by nodes with imprecise probabilities, and we obtain a credal network where we maximize the sum of the marginal probabilities of the utility nodes. The objective function is

$$\max\ p(COA) + p(COB) + p(POG)$$

(here we use the notation $p(\cdot)$ instead of $f(\cdot)$ because we are treating the utility nodes after translating them into probability nodes) subject to constraints that define each marginal probability $p(COA)$, $p(COB)$ and $p(POG)$. To create these constraints, we run a symbolic Bayesian network inference for each of them. The constraints for $p(COA)$ and $p(COB)$ are very simple:

$$p(COA) = p(COA|\text{DGA=1})p(\text{DGA=1}) + p(COA|\text{DGA=0})p(\text{DGA=0}),$$

$$p(COB) = p(COB|\text{BB=1})p(\text{BB=1}) + p(COB|\text{BB=0})p(\text{BB=0}),$$

because they only depend on one other variable. Note that $p(\text{DGA=1})$, $p(\text{DGA=0})$, $p(\text{BB=1})$, and $p(\text{BB=0})$ that appear in these constraints are unknown and thus become optimization variables in the bilinear problem.

To write the constraints for $p(POG)$, we need to choose a precedence ordering. We will use the ordering BB, BC, DGA, GA, TO, POG (variables COA and COB do not appear in the order as they are not relevant to evaluate the marginal $p(POG)$). Hence, the first variable to be processed is BB. We write a constraint that relates the query POG and probabilities $p(BB)$ (which are defined in the network specification):

$$p(POG) = \sum_{d\in\{0,1\}} p(\text{BB} = d) \cdot p(POG|\text{BB} = d).$$

BB now appears in the conditional part of $p(POG|d)$, which may be viewed as an artificial term in the optimization, as it does not appear in the network. Because of that, we must create constraints to define $p(POG|d)$ in terms of network parameters (for all categories $d \in BB$). According to our chosen ordering, the current variable to be processed is BC. Thus,

$$\begin{aligned} p(POG|\text{BB=1}) &= \sum_{c\in\{0,1\}} p(\text{BC} = c|\text{BB=1}) \cdot p(POG|\text{BC} = c), \\ p(POG|\text{BB=0}) &= \sum_{c\in\{0,1\}} p(\text{BC} = c|\text{BB=0}) \cdot p(POG|\text{BC} = c). \end{aligned}$$

Note that $p(POG|c) = p(POG|c, d)$ (for any d), so we use the simpler. At this stage, our query is conditioned on BC. Following the same idea, we process DGA, obtaining

$$p(POG|\text{BC=1}) = \sum_{d\in\{0,1\}} p(DGA = d) \cdot p(POG|\text{BC=1}, DGA = d),$$

$$p(POG|\text{BC=0}) = \sum_{d\in\{0,1\}} p(DGA = d) \cdot p(POG|\text{BC=0}, DGA = d).$$

Now the current variable to be treated is GA, and our query is conditioned on BC, DGA, that is, we must define how to evaluate $p(POG|BC, DGA)$ for all configurations. Thus, for all $c \in \{0, 1\}$ and $d \in \{0, 1\}$:

$$p(POG|BC = c, DGA = d) = \sum_{c'\in\{0,1\}} p(GA = c'|BC = c, DGA = d) \cdot p(POG|BC = c, GA = c').$$

At this moment, POG is conditioned on GA, BC in the artificial term $p(POG|BC = c, GA = c')$ (DGA is not present in the artificial term as GA, BC separate POG from DGA). Now we process TO: for all $c' \in \{0,1\}$ and $c \in \{0,1\}$

$$p(POG|BC = c, GA = c') = \sum_{c'' \in \{0,1\}} p(TO = c''|BC = c, GA = c') \cdot p(POG|TO = c'').$$

Note that, as $p(POG|TO = c'')$ equals the utility function of POG given TO, which is specified in the network, we can stop the symbolic elimination. All artificial terms are related (through constraints) to parameters of the network. Besides all these constraints, we also include simplex constraints to ensure that probabilities sum 1. Finally, we need to include the constraints of Equations (3) and (5) (and other utility constraints that we omitted for easy of expose, as explained in Section 3.1). To illustrate using the same notation, we rewrite the Equations (5) here:

$$-90 \le I_{l_{\mathrm{DGA}}}(1) \cdot r \cdot p(COA|DGA = 1) + I_{l_{\mathrm{DGA}}}(0) \cdot r \cdot p(COA|DGA = 0) \le -60,$$

$$0 \le I_{l_{\mathrm{DGA}}}(1) \cdot r \cdot p(COA|DGA = 1) + I_{l_{\mathrm{DGA}}}(0) \cdot r \cdot p(COA|DGA = 0) \le 0,$$

and

$$I_{l_{\mathrm{DGA}}}(1) + I_{l_{\mathrm{DGA}}}(0) = 1,$$

where r is the maximum utility value that appears in the whole database.

Because we have a collection of linear and bilinear constraints, non-linear programming can be employed [3]. It is also possible to use linear integer programming [4] after an additional manipulation of the constraints.

4 Conclusion

In this work we have presented a complete learning procedure for LIMIDs. To deal with chance nodes, a standard technique from machine learning is employed. To estimate utility functions, we make use of a database containing past decisions and rewards, which avoids the necessity of interviews with experts to elict the utilities. With this approach, it is expected that the model represents the corresponding domain with more precision. The procedure creates a set of constraints that define the plausible utility functions, which can be later processed by the strategy selection method. Hence, this work provides a learning procedure to be integrated with the inference methods that we have developed in the previous years. Altogether, they provide a concise solution for military planning using the general Limited Memory Influence Diagrams as basis.

Future work include the full integration of the learning procedure described here with the inference procedures already developed, extension of these ideas to structure discovery of utilities [2], and the exploration of new learning and strategy methods. For instance, it is known that both learning and inference methods can be performed very efficiently (polynomial time) in tree-shaped diagrams. We have already started to study a broader version of influence diagrams that extends the trees but are more restricted than LIMIDs. It is mainly a type of LIMID where decision nodes are related by a tree structure, while chance nodes are free to happen anywhere. The advantages of such intermediate model is the possibility of keeping the computations tractable (we have proved that LIMIDs can be processed up to a couple of hundreds of nodes, but even larger domains are much more time consuming and need efficient algorithms).

References

[1] U. Chajewska and D. Koller and D. Ormoneit. Learning an Agent's Utility Function by Observing Behavior. In International Conference on Machine Learning, 35–42, 2001.

[2] U. Chajewska and D. Koller. Utilities as random variables: Density estimation and structure discovery. In Conference on Uncertainty in Artificial Intelligence, 63–71, 2000.

[3] C. P. de Campos and F. G. Cozman. Inference in credal networks using multilinear programming. In *Second Starting AI Researcher Symposium*, p. 50–61, Valencia, 2004. IOS Press.

[4] C. P. de Campos and F. G. Cozman. Inference in credal networks through integer programming. In *Int. Symp. on Imprecise Probability: Theories and Applications*, p. 145–154, 2007.

[5] C. P. de Campos and Q. Ji. Strategy Selection in Influence Diagrams using Imprecise Probabilities. In Conference on Uncertainty in Artificial Intelligence, 121–128, 2008.

[6] D. G. Fromberg and R. L. Kane. Methodology for measuring health-state preferences – II: Scaling methods Journal of Clinical Epidemiology, 42:459–471, 1989.

[7] R. A. Howard and J. E. Matheson. *Influence diagrams*, volume II, p. 719–762. Strategic Decisions Group, Menlo Park, 1984.

[8] S. Lauritzen and D. Nilsson. Representing and solving decision problems with limited information. *Management Science*, 47:1238–1251, 2001.

[9] A. Y. Ng and S. Russell. Algorithms for Inverse Reinforcement Learning. In International Conference on Machine Learning, 663–670, 2000.

[10] T. D. Nielsen and F. V. Jensen. Learning a decision maker's utility function from (possibly) inconsistent behavior. Artificial Intelligence, 160(1-2), 53–78, 2004.

[11] M. Zaffalon. Conservative rules for predictive inference with incomplete data. In International Symposium on Imprecise Probabilities and Their Applications, 406-415, 2005.

Validation of Cassio's software

Geng Li

1. Introduction

In this first part of this report, I provided three test examples used for the validation of Cassio's software. The first example is randomly created from the reality, the second example comes from the high cited paper "Representing and Solving Decision Problems with Limited Information" and the third example is a small EBO net from the project.

In the second part of this report, I implemented Brutal Force method based on BNT to enumerate all possible strategies in Cassio's example and computed the expected utility for all the strategies accordingly. Then I compared the results to the outcomes by SPU and CR. All tests proved that Cassio's method will converge to global maximum solution, and SPU will only converge to local maximum.

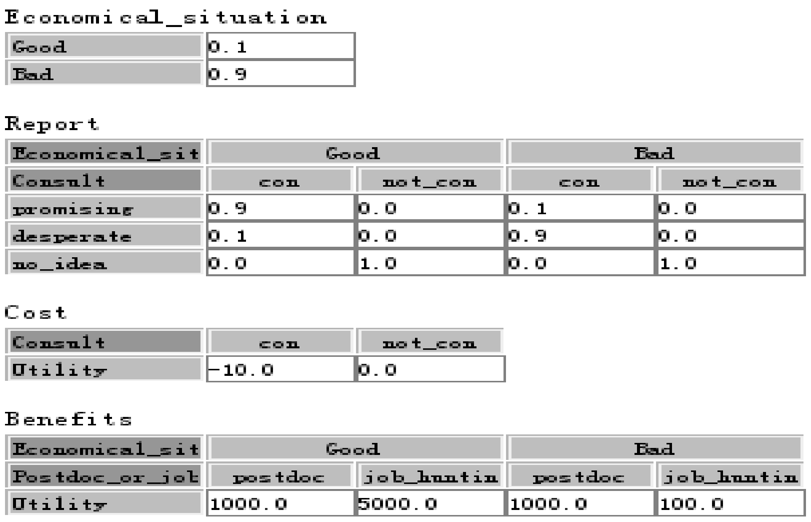
2. Validation of the Implementation of the software

For Cassio's software, it right now could deal with LIMIDs version of the diagram. For the *GeNIe*(A software developed by University of Pittsburgh), it could handle the traditional ID. So the reason why Cassio's example could not be validated in *GeNIe* is that *the order of decision nodes in the example is not specified.* Additionally, if we want to test Cassio's example in *GeNIe*, *we have to explicitly make arcs from predecessors of decision nodes. It will create too many links in the large networks and it is not operable in reality.* So we could not use *GeNIe* that could only handle traditional ordered IDs to handle Cassio's EBO example which is a LIMIDs with no specified order. But we could test a certain LIMIDs that has specified order and explicit links. Because by explicitly making arcs, a certain diagram could be viewed as either a tradition influence diagram or a LIMIDs. They would get the same result. This is what I did in the previous report. The three examples are showed in details as follows:

(1) Example 1

EX1: John now is close to graduate. Now he has two choices, one is to continue his postdoc research and the other is job-hunting. But before making his decisions, he wants to refer to the economical situation and thus consult to the specialist. The consulting fee is about 10 dollars. If the economical situation is good, the probability that the specialist makes a promising prediction is 0.9 and if bad, the probability that the specialist makes a promising prediction is 0.1. The benefit of continuing postdoc is 1000 dollars no matter what current situation is. However, if the economical situation is good, the average benefit of getting a job is 5000 dollars and if the economical situation is bad, the average benefit of getting a job is only 100 dollars.

The CPDs for the example are displayed in Figure 1, and the diagrams established by Cassio's software and GeNIe are showed in Figure 2 and Figure 3 respectively:



Economical_situation

Good	0.1
Bad	0.9

Report

Economical_sit	Good		Bad	
Consult	con	not_con	con	not_con
promising	0.9	0.0	0.1	0.0
desperate	0.1	0.0	0.9	0.0
no_idea	0.0	1.0	0.0	1.0

Cost

Consult	con	not_con
Utility	-10.0	0.0

Benefits

Economical_sit	Good		Bad	
Postdoc_or_job	postdoc	job_huntin	postdoc	job_huntin
Utility	1000.0	5000.0	1000.0	100.0

Figure 1 CPDs for example 1

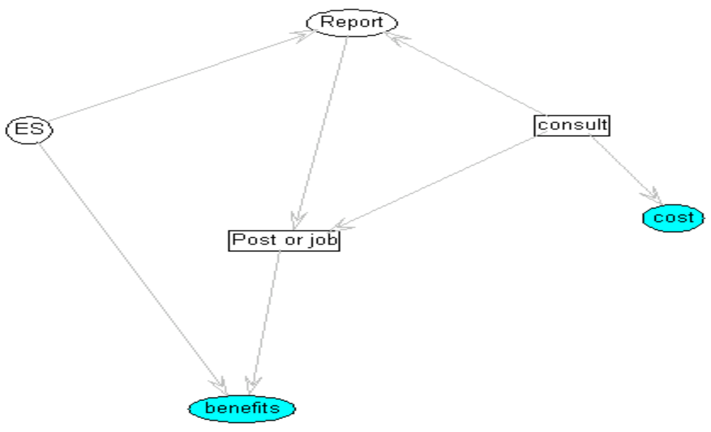


Figure 2 The diagram created by Cassio's software

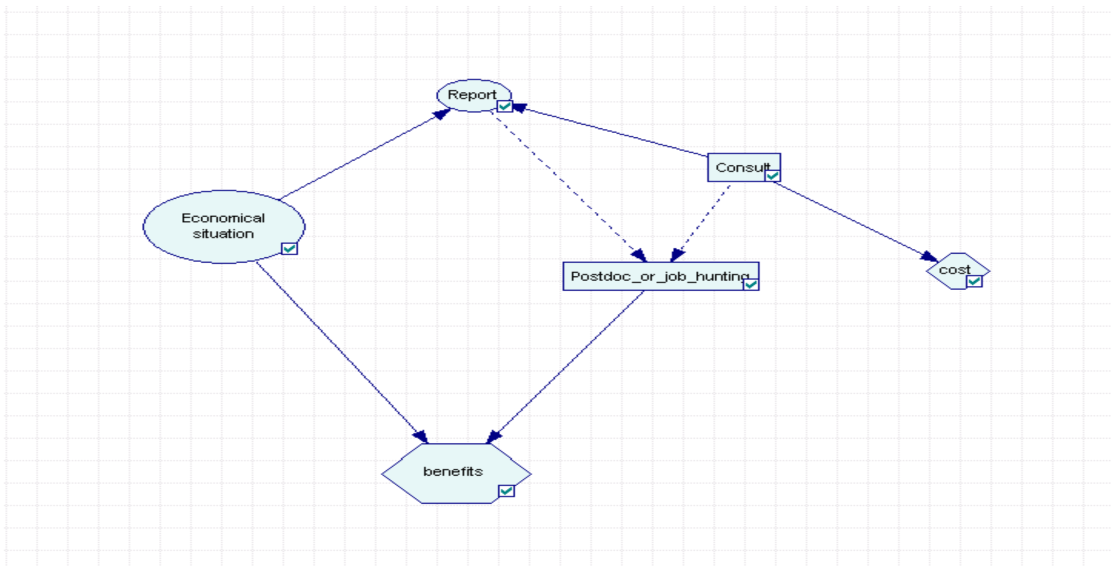
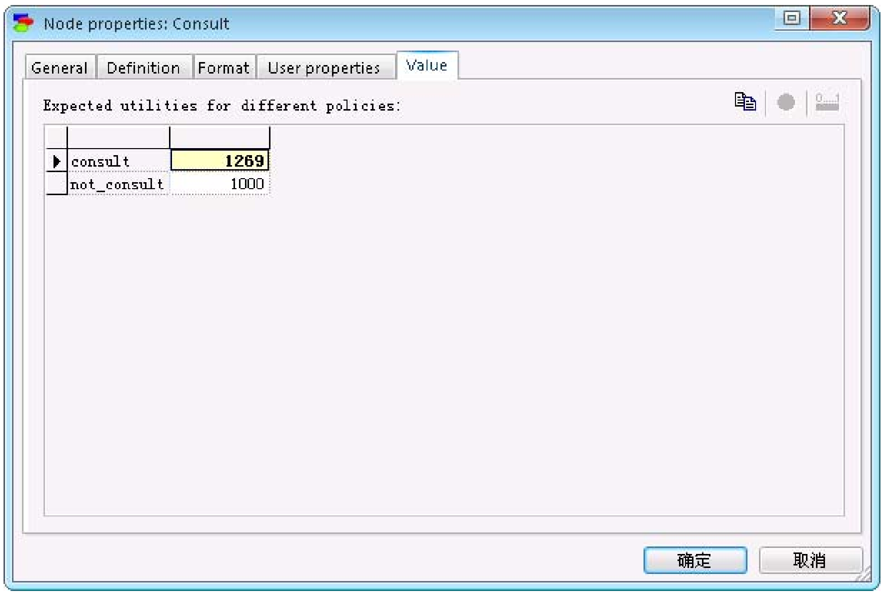


Figure 3 The diagram created by GeNIe

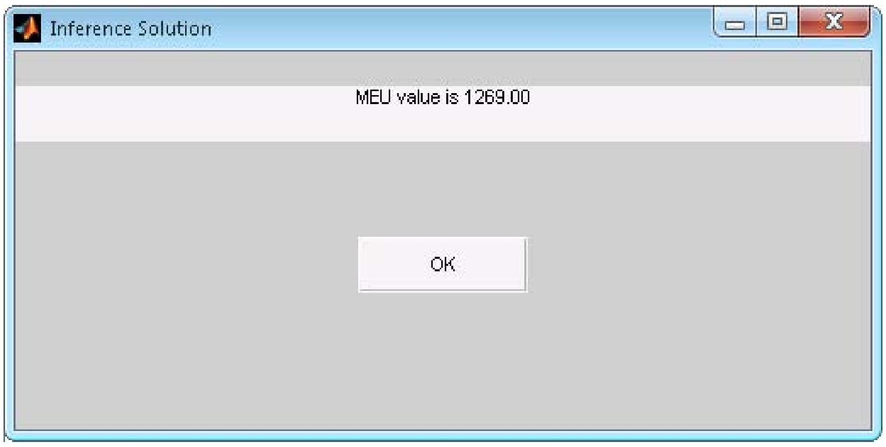
(i)Software validation

The calculated MEU in both softwares are showed in Figure 4, we see that both softwares get the same MEU 1269(also in Cassio's software, two methods SPU and CR got the same result). In Figure 5, the optimal strategies selected in two softwares are the same as well. Note that in GeNIe, the bold-faced numbers denotes the optimal action to take given its configuration of parents. In Cassio's software, the green rectangle denotes the optimal policy to take.



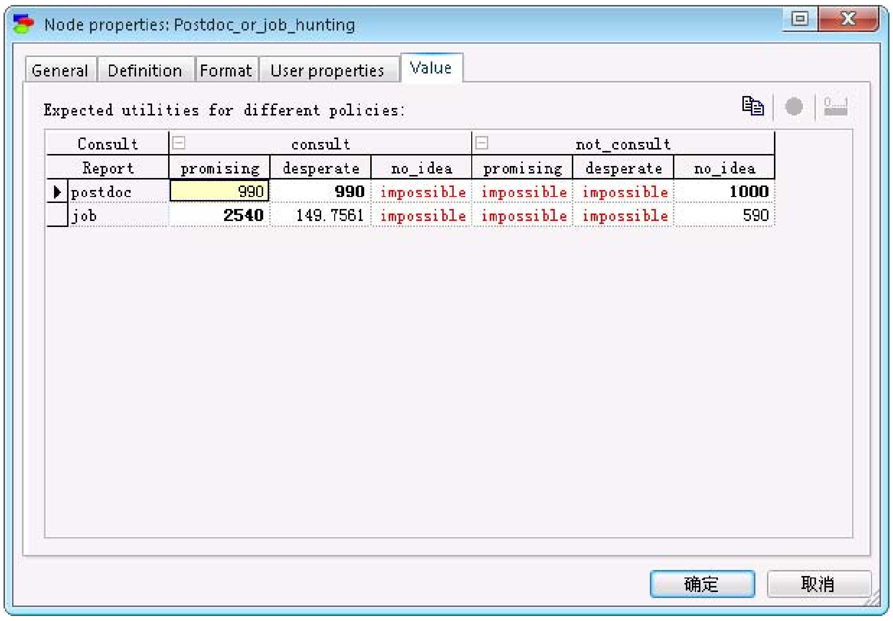
Expected utilities for different policies:

consult	**1269**
not_consult	1000



MEU value is 1269.00

Figure 4 The upper result is from GeNIe, and the lower result is from Cassio's software



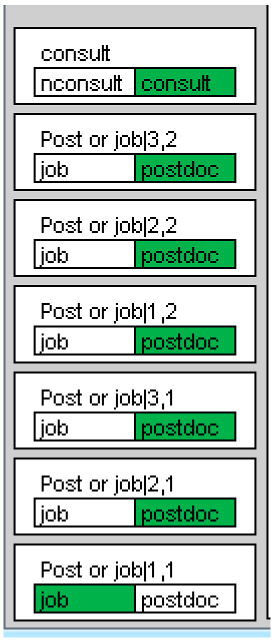
Node properties: Postdoc_or_job_hunting

General | Definition | Format | User properties | Value

Expected utilities for different policies:

Consult	consult			not_consult		
Report	promising	desperate	no_idea	promising	desperate	no_idea
postdoc	990	990	impossible	impossible	impossible	1000
job	2540	149.7561	impossible	impossible	impossible	590

确定 取消



consult
nconsult [✓ consult]

Post or job|3,2
job [✓ postdoc]

Post or job|2,2
job [✓ postdoc]

Post or job|1,2
job [✓ postdoc]

Post or job|3,1
job [✓ postdoc]

Post or job|2,1
job [✓ postdoc]

Post or job|1,1
[✓ job] postdoc

Figure 5 The optimal strategies from the two softwares

(ii)Manual Calculation

Now I shall manually calculate the above example and validate the software. The expected utility of the whole diagram is calculated using:

$$EU(D_C)=\sum_{R}\max_{D_{P\_or\_J}}\sum_{ES}P(ES)P(R\mid ES,D_C)\{U_{\text{cost}}(D_C)+U_2(ES,D_{P\_or\_J})$$

The main of the SPU is that keeping all other policies unchanged and we only calculate one policy each time to get a new MEU. Then we replace the new optimal policy with the original one.

We initially assume that the original strategies are consult and postdoc. According to SPU algorithm, for the decision node "Postdoc_or_Job_hunting":

$$EU(D_{P\_or\_J} = 'Job' \mid D_C = 'consult', R = 'promi\sin g')$$
$$= \sum_{ES} P(ES)P(R = 'promi\sin g' \mid ES, D_C = 'consult')\{U_{\cos t}(D_C = 'consult') + U_{Benefit}(ES, D_{P\_or\_J} = 'Job')\}$$
$$= 0.1*0.9*\{-10+5000\} + 0.9*0.1*\{-10+100\}$$
$$= 457.2$$

Similarly, we calculated other expected utility of the decision node "Post_or_job" considering each of its parent's configurations:

$$EU(D_{P\_or\_J} = 'Job' \mid D_C = 'consult', R = 'desperate') = 121.9$$
$$EU(D_{P\_or\_J} = 'Job' \mid D_C = 'consult', R = 'no\_idea') = impossible$$

$$EU(D_{P\_or\_J} = 'Post' \mid D_C = 'consult', R = 'promi\sin g')$$
$$= \sum_{ES} P(ES)P(R = 'promi\sin g' \mid ES, D_C = 'consult')\{U_{\cos t}(D_C = 'consult') + U_{Benefit}(ES, D_{P\_or\_J} = 'Post')\}$$
$$= 0.1*0.9*\{-10+1000\} + 0.9*0.1*\{-10+1000\}$$
$$= 178.2$$
$$EU(D_{P\_or\_J} = 'Post' \mid D_C = 'consult', R = 'desperate') = 811.8$$
$$EU(D_{P\_or\_J} = 'Post' \mid D_C = 'consult', R = 'no\_idea') = impossible$$

Because if $D_{P\_or\_J} = 'Post'$ no matter what the result of R is, the EU of the strategies is $178.2 + 811.8 = 990.0$, which is less than $457.2 + 811.8 = 1269.0$. Thus given the decision consult for the decision node "Consult", if the "Report" is promising, the policy for the node "Post_or_job" is to go and find a job. Otherwise, the policy for the node "Post_or_job" will remain unchanged. We update our policy and now we could use this result to calculate the expected utility of the decision node "Consult".

$$EU(D_C = 'Consult')$$
$$= \sum_{R} \max_{D_{P\_or\_J}} \sum_{ES} P(ES)P(R \mid ES, D_C)\{U_{\cos t}(D_C) + U_2(ES, D_{P\_or\_J})$$
$$= 457.2 + 811.8 = 1269.0$$

Similarly,

$$EU(D_C = 'not\_Consult') = 0.1*1*(0+1000) + 0.9*1*(0+1000) = 1000$$

Thus the optimal policy for D_C remains unchanged, that is, to consult. For the second round calculation using SPU method, the strategies would not change because EU will no longer change. Therefore, $MEU = \max_{D_C} EU(D_c) = 1269.0$

<u>From above, we calculated the MEU for the decision node "Consult", which is also the MEU for the whole diagram, is 1269.0. Thus, we see that Cassio's method got the correct result.</u>

(2) Example 2

Ex 2. Another example comes from the paper "Representing and Solving Decision Problems with Limited Information" by Steffen L. Lauritzen and Dennis Nilsson in 2001. Due to the example already mentioned in my previous report, I will only present the graph model and the calculated result here.

Figure 6 shows the graph created by Cassio's software. Here I replaced the original code by the updated version from the latest Cassio's email. The calculated MEU by SPU and CR is shown is figure 7 and we see that Cassio's method gets the same result with the paper "Representing and Solving Decision Problems with Limited Information" in Figure 8.

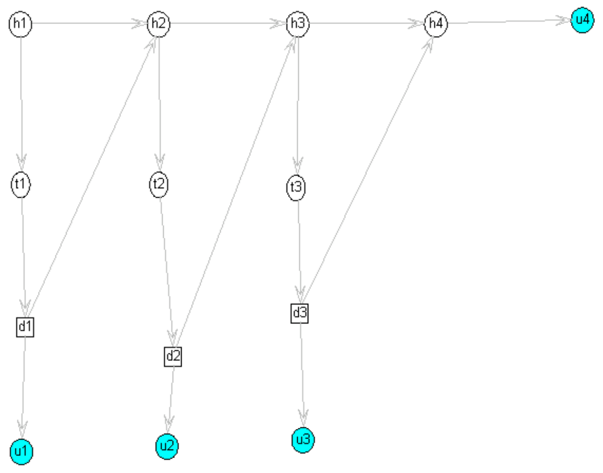
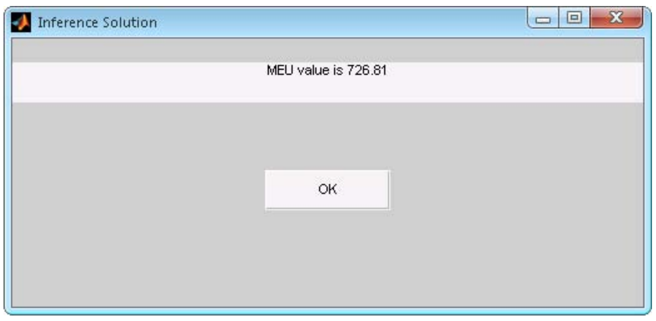


Figure 6 The Pigs model created using Cassio's software



MEU value is 726.81

Figure 7 The calculated MEU from Cassio's software

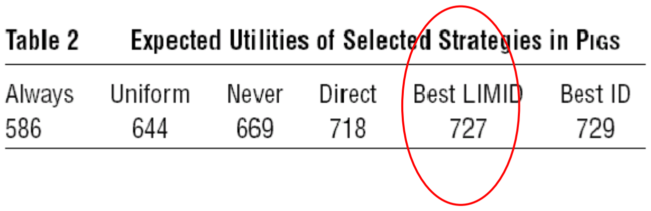


Table 2 Expected Utilities of Selected Strategies in Pigs

Always	Uniform	Never	Direct	Best LIMID	Best ID
586	644	669	718	727	729

Figure 8 The result from the cited paper

(3) Example 3

Ex3. In Figure 9, I manually created a median-size network which is similar to Cassio's example of EBO net. The reason why I created the median network because this free software, Hugin, has the limitation of the number of the nodes. But anyway, the two software got the same MEU result. Figure 10 shows the calculated MEU is 790.89 by using Cassio's software from SPU and CR methods and Figure 11 shows the same result, that is, MEU equals to 790.89.

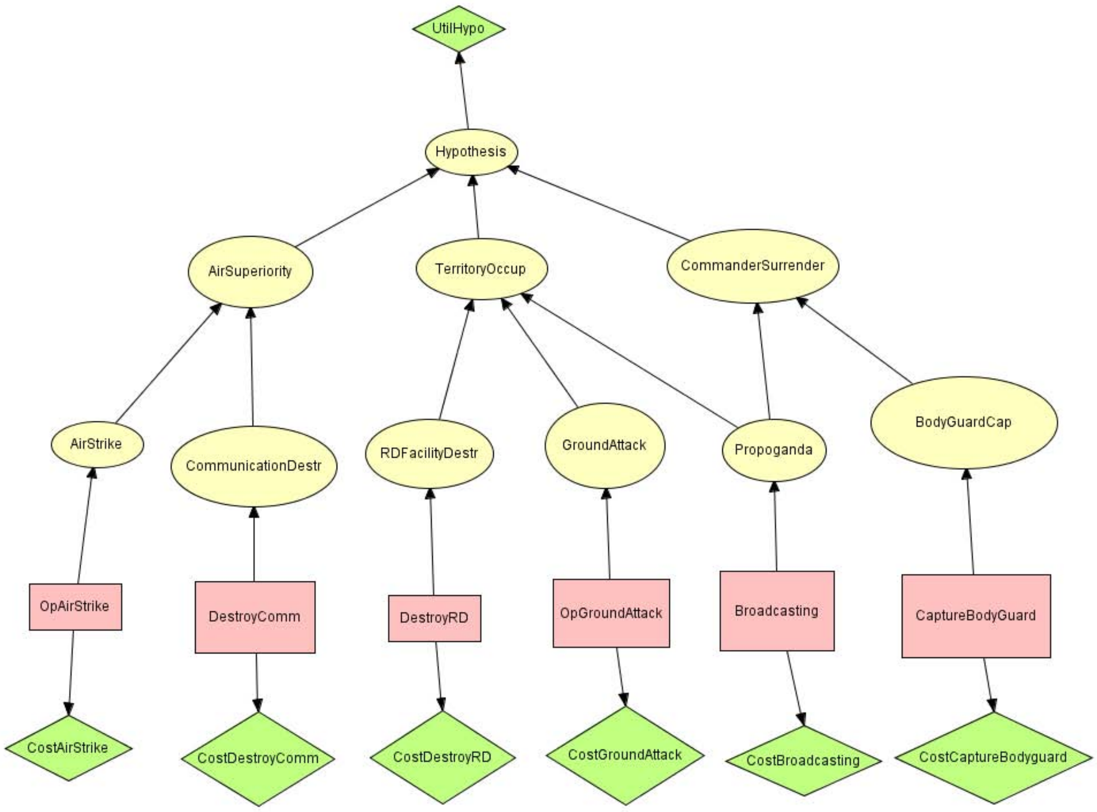


Figure 9 The diagram created by Hugin

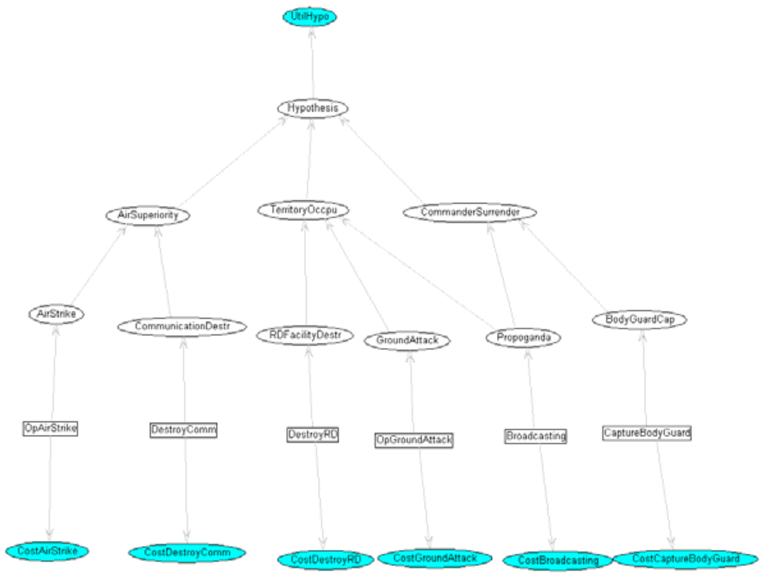
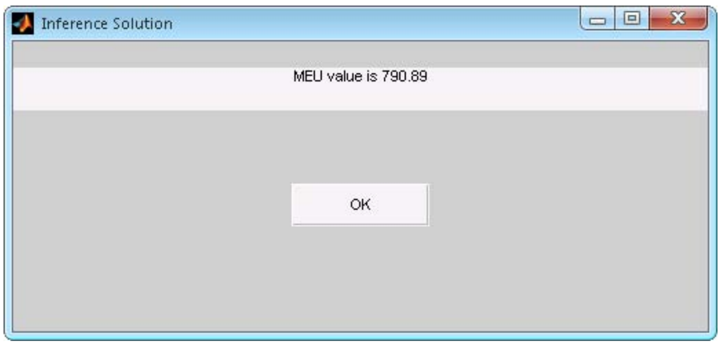
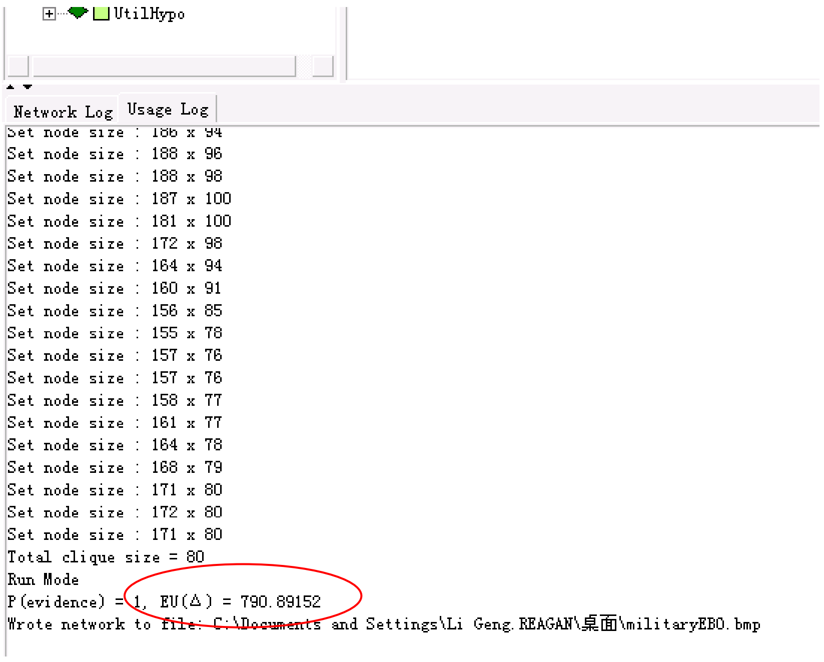


Figure 10 The diagram created by Cassio's software



MEU value is 790.89

Figure 11 MEU calculated using Cassio's software



```
Set node size : 186 x 94
Set node size : 188 x 96
Set node size : 188 x 98
Set node size : 187 x 100
Set node size : 181 x 100
Set node size : 172 x 98
Set node size : 164 x 94
Set node size : 160 x 91
Set node size : 156 x 85
Set node size : 155 x 78
Set node size : 157 x 76
Set node size : 157 x 76
Set node size : 158 x 77
Set node size : 161 x 77
Set node size : 164 x 78
Set node size : 168 x 79
Set node size : 171 x 80
Set node size : 172 x 80
Set node size : 171 x 80
Total clique size = 80
Run Mode
P(evidence) = 1, EU(Δ) = 790.89152
Wrote network to file: C:\Documents and Settings\Li Geng.REAGAN\桌面\militaryEBO.bmp
```

Figure 12 MEU calculated using Hugin

3. Comparison of SPU CR

In this part, I implemented a Brutal Force method which enumerates all possible strategies and got all the EU results.

(1)Original example

In Cassio's example in Figure 13, there are totally 11 decision nodes. Each node has two choices, that is, "take" and "do not take". So the number of all possible strategies is $2^{11} = 2048$. My program will traverse all these 2048 strategies and selected the largest ten MEU.

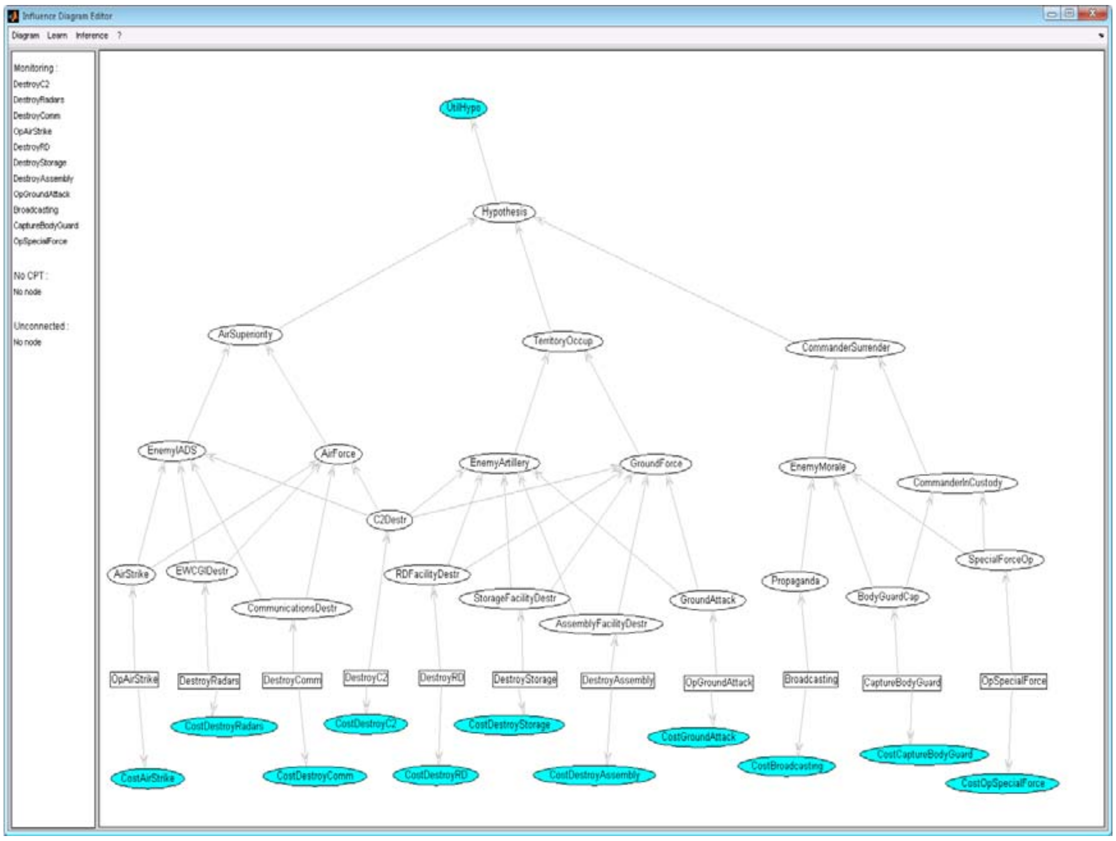
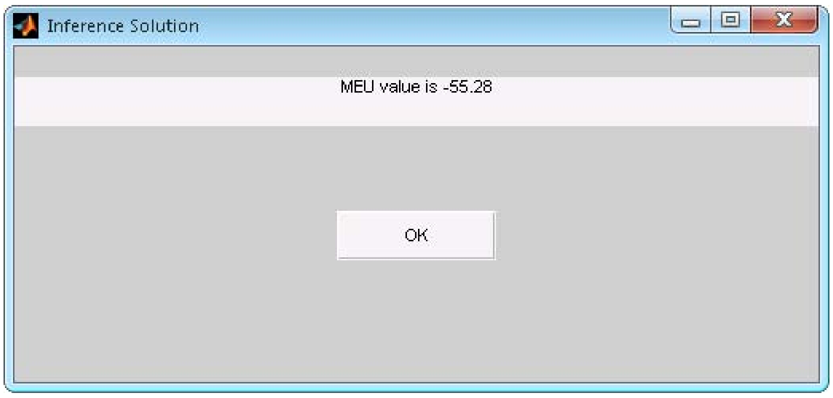


Figure 13 Military example

In the original example provided by Cassio and already specified parameters, the largest ten EU calculated by brutal force method are shown in Table 1. Also I ran the software and the MEU calculated by CR and SPU are 156.4 and -55.28 respectively that are shown in Figure 14 and Figure 15. From the table, we clearly see that the result of CR method is exactly the same with the first item of the table. This proves that in this example CR method converged to the global maximum. Also the result of SPU matches the ninth item of the table which denotes that SPU will only converge to the local maximum.

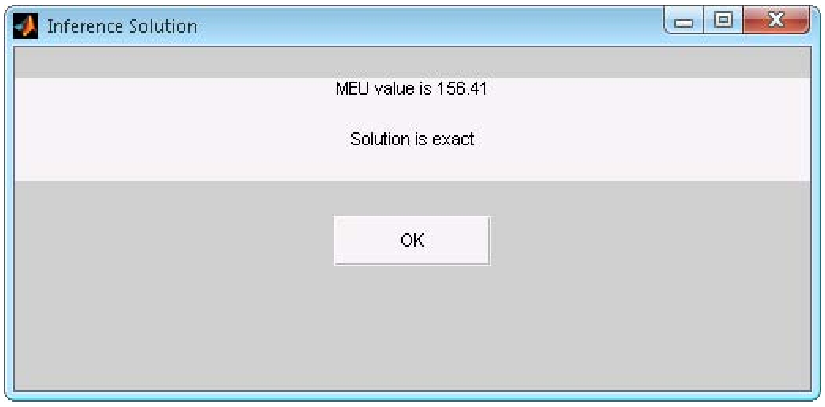
NO.	1	2	3	4	5	6	7	8	9	10
MEU	156.4	68.97	55.26	12.89	-7.11	-20.59	-34.21	-51.26	-55.28	-61.03

Table 1 The largest ten EU in Cassio's example



MEU value is -55.28

Figure 14 SPU result


Inference Solution

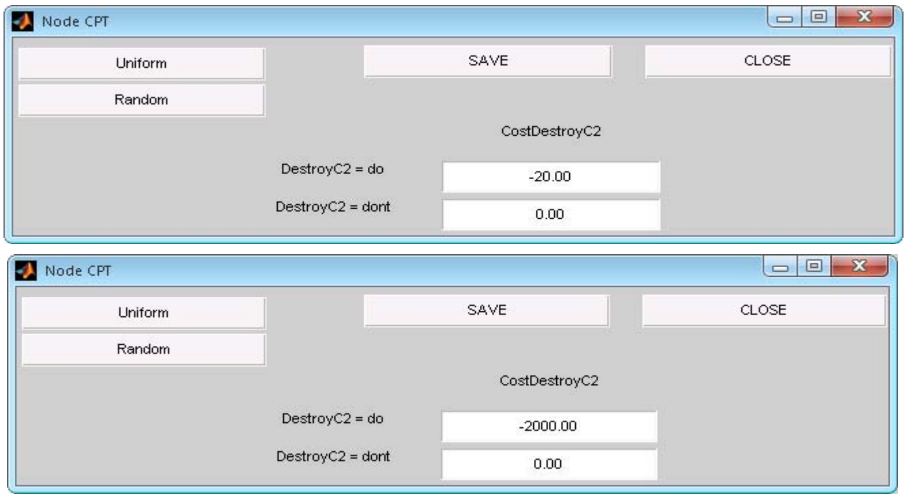
MEU value is 156.41

Solution is exact

Figure 15 CR result

(2) Change the utility value of the node "CostDestroyC2"

In the second test, I changed the utility value of the node "CostDestroyC2" from [-20, 0] to [-2000, 0]. This is shown in Figure 16.



CostDestroyC2

DestroyC2 = do -20.00

DestroyC2 = dont 0.00

CostDestroyC2

DestroyC2 = do -2000.00

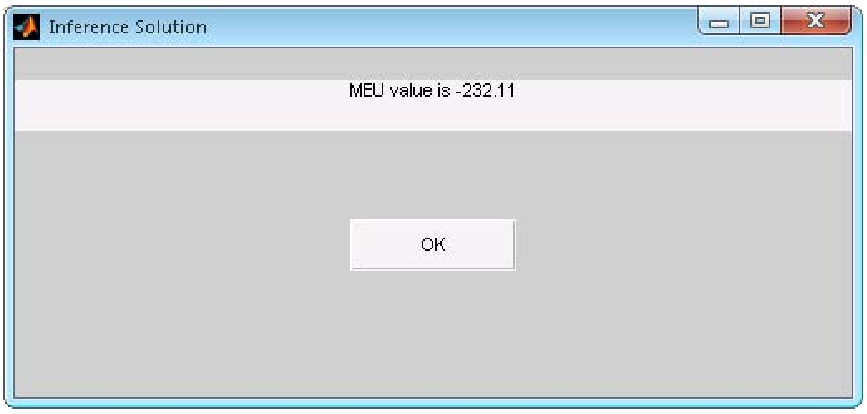
DestroyC2 = dont 0.00

Figure 16 The utility value of the node "CostDestroyC2"

Then I conducted brutal force method for this example and the largest ten EU results are in shown in table 2. Note that different strategies could lead to the same MEU. The result by SPU and CR are shown in Figure 17. This time they both got the same result, MEU -232.11. Referring to table 2, this is reasonable because both of them converge to the global maximum.

NO.	1	2	3	4	5	6	7	8	9	10
MEU	-232.1	-252.1	-252.1	-252.1	-272.1	-272.1	-278.8	-292.1	-306.1	-312.3

Table 2 The largest ten EU

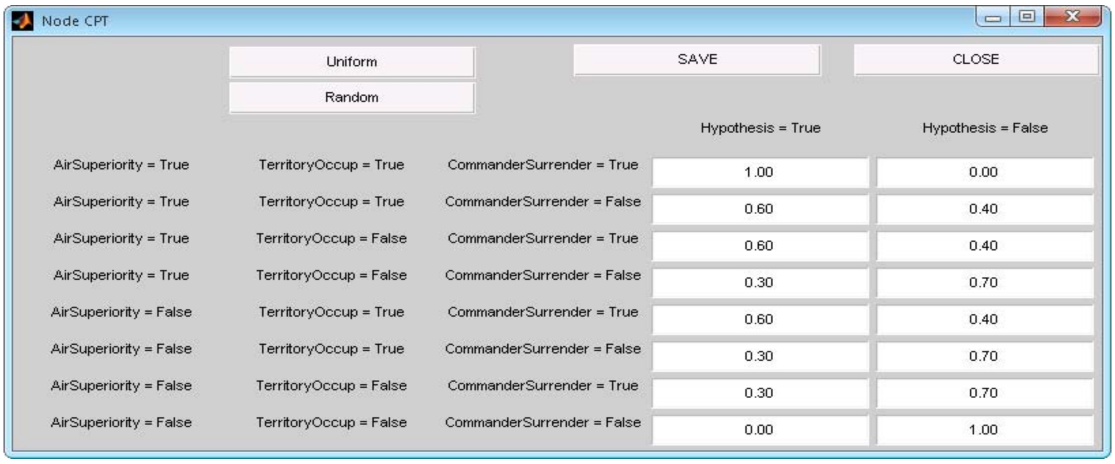

Inference Solution

MEU value is -232.11

Figure 17 the result of SPU and CR

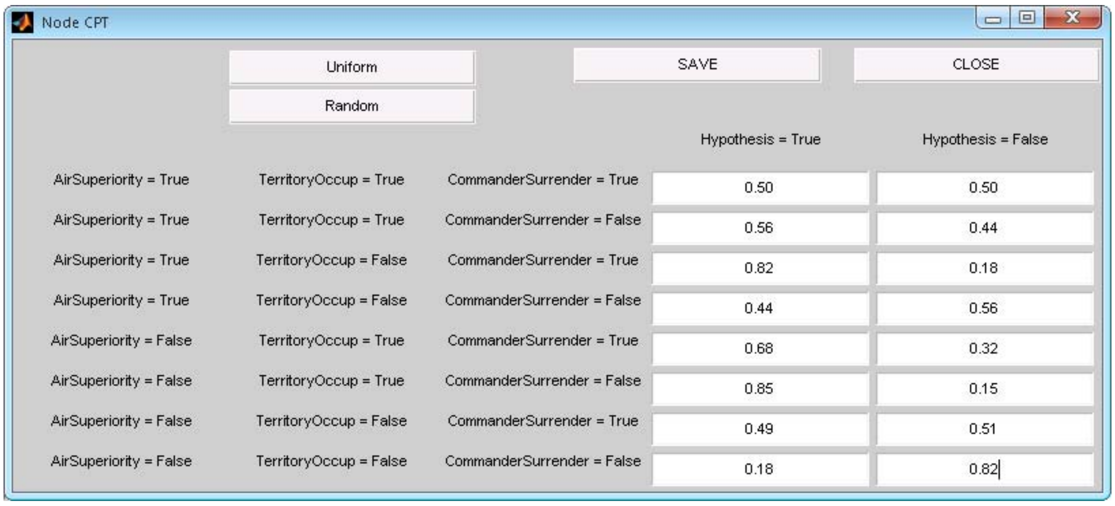
(3) Change CPT of the chance node “Hypothesis”

In the third test, the CPTs of the chance node “Hypothesis” before and after the change are shown in Figure 18 and Figure 19.


Node CPT

			Hypothesis = True	Hypothesis = False
AirSuperiority = True	TerritoryOccup = True	CommanderSurrender = True	1.00	0.00
AirSuperiority = True	TerritoryOccup = True	CommanderSurrender = False	0.60	0.40
AirSuperiority = True	TerritoryOccup = False	CommanderSurrender = True	0.60	0.40
AirSuperiority = True	TerritoryOccup = False	CommanderSurrender = False	0.30	0.70
AirSuperiority = False	TerritoryOccup = True	CommanderSurrender = True	0.60	0.40
AirSuperiority = False	TerritoryOccup = True	CommanderSurrender = False	0.30	0.70
AirSuperiority = False	TerritoryOccup = False	CommanderSurrender = True	0.30	0.70
AirSuperiority = False	TerritoryOccup = False	CommanderSurrender = False	0.00	1.00

Figure 18 The CPT of “Hypothesis” before the change


Node CPT

			Hypothesis = True	Hypothesis = False
AirSuperiority = True	TerritoryOccup = True	CommanderSurrender = True	0.50	0.50
AirSuperiority = True	TerritoryOccup = True	CommanderSurrender = False	0.56	0.44
AirSuperiority = True	TerritoryOccup = False	CommanderSurrender = True	0.82	0.18
AirSuperiority = True	TerritoryOccup = False	CommanderSurrender = False	0.44	0.56
AirSuperiority = False	TerritoryOccup = True	CommanderSurrender = True	0.68	0.32
AirSuperiority = False	TerritoryOccup = True	CommanderSurrender = False	0.85	0.15
AirSuperiority = False	TerritoryOccup = False	CommanderSurrender = True	0.49	0.51
AirSuperiority = False	TerritoryOccup = False	CommanderSurrender = False	0.18	0.82

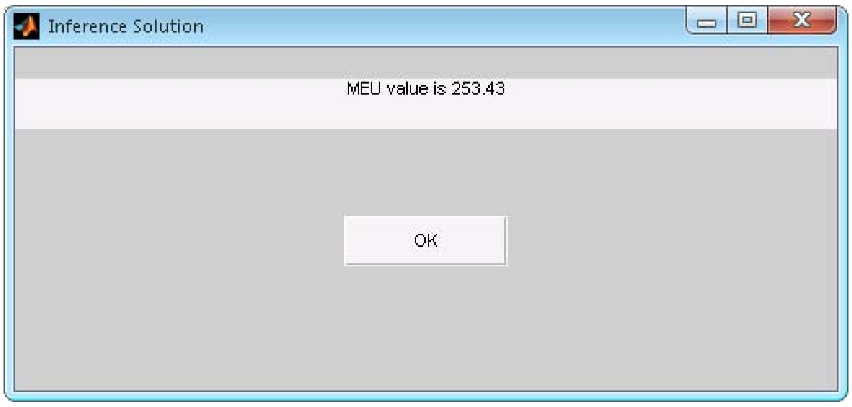
Figure 19 The CPT of “Hypothesis” after the change

Using Brutal Force algorithm, the calculated ten largest MEU are shown in table 3. The calculated

MEU by SPU and CR method are shown in Figure 20 and Figure 21. CR method got the largest MEU 298.3 and SPU only got MEU 253.4. From the result, we see that again CR method will converge to the global maximum and SPU will only converge to the local maximum.

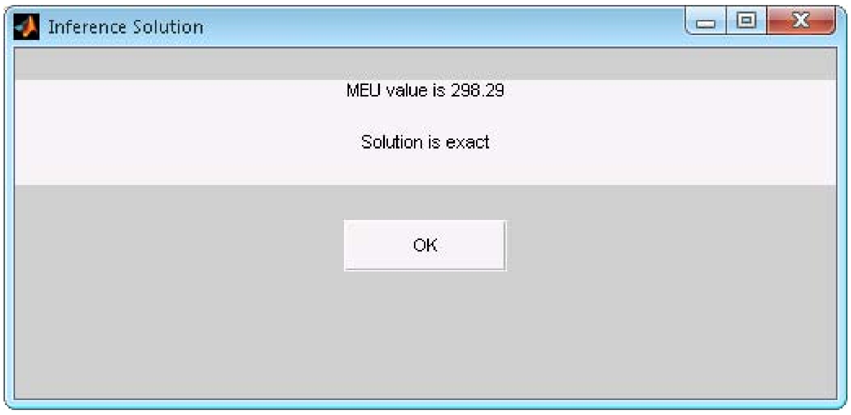
NO.	1	2	3	4	5	6	7	8	9	10
MEU	298.3	278.3	278.3	278.3	258.3	258.3	253.4	248.3	233.4	233.4

Table 3 The largest ten EU



MEU value is 253.43

Figure 20 SPU method



MEU value is 298.29

Solution is exact

Figure 21 CR method

4. Conclusion

In the first part of this report I created three examples and used them to validate Cassio's software. The calculated results are compared to the existing software and the paper. We conclude that Cassio's implementation got all the correct results.

In the second part of the report, we conclude that CR performs better than SPU because it will converge to the global maximum solution. But, SPU is a good approximate method and for many networks, it is expected that it will achieve the global optimal solution as well. But it is not always the case, just as shown in the first and third test of this part. SPU is some kind of like Nash Equilibrium, which means that all its policies are local maximum. Here, a strategy is a local maximum strategy means that the expected utility does not increase by changing only one of its policies. Thus, SPU will only guarantee to converge to the local maximum solution.

Learning Influence Diagram Parameters using Convex Optimization for EBO-based Planning

ISL
Rensselaer Polytechnic Institute
Troy, NY

December 21, 2007

Abstract

An Influence Diagram is an interesting probabilistic graphical framework for modeling effects-based operations (EBO), as they provide a compact representation of the domain and useful properties for decision making. It is known that accuracy of decisions relies on the quality of Influence Diagram parameters. On the one hand, learning reliable parameters of such models often requires a relative large amount of quantitative training data, which may not exist, may be hard to acquire and/or may contain missing values. On the other hand, qualitative knowledge information is usually available, and incorporating such knowledge can improve parameter estimation and the accuracy of decisions. This report describes a framework based on convex optimization to incorporate many types of qualitative relations about parameters with quantitative training data to perform parameter estimation in an Influence Diagram for the EBO planning problem. Experiments and examples using synthetic data indicate the benefits of this framework.

1 Introduction

The Effects-based operations (EBO) approach for military planning seeks for a campaign objective by considering direct, indirect and cascading effects of military, diplomatic, psychological and economic actions [6, 9]. Graphical models such as Influence Diagrams are specially interesting while modeling such a domain, as we can specify many actions and factors, uncertainties and their dependencies. We employed a graphical model called Influence Diagram, which comprises decision, uncertainty and utility information in a compact graphical structure for decision making [11, 19, 23].

Influence Diagrams are described through graphs where value nodes are related to profits and costs of actions, chance nodes represent uncertainties and dependencies in the domain and decision nodes represent actions to be taken. After parametrization, this model can be used to evaluate plans (expected utility) and strategy decisions (choice of which action to take). Parameter learning is the problem of estimating probability measures of conditional probability distributions related to the chance nodes, given the graph structure.

Many parameter learning techniques depend heavily on training data. Ideally, with sufficient data, it is possible to learn parameters by standard statistical analysis like maximum likelihood estimation. In many real-world cases, however, data are either incomplete or sparse, which can cause inaccurate parameter estimation. Incompleteness means that some parameter values are missing in the data, while sparseness means that the amount of training data is small. EBO military planning is a domain with such characteristics, as the world history has not too many military campaigns. Furthermore, data about these conflicts may be sensitive/secret and may not be available.

Even with incomplete and sparse data, general qualitative knowledge about the domain is usually available, and such knowledge might be employed to improve parameter estimations. While the previous report [12] has focused mainly on model evaluation and has supposed that parameters were known in advance, we propose here a framework based on non-linear convex optimization to solve probabilistic parameter learning problem by combining quantitative data and domain knowledge in the form of qualitative constraints. Hence, this report is complementary to [12] in the sense that parameter learning is the main focus.

Many types of qualitative constraints are discussed, including range and relationship constraints [16], influences and synergies [25, 26], non-monotonic constraints [24], weak and strong qualitative constraints [4, 20]. They are applied to an EBO-based planning problem and experiments indicate that qualitative constraints can strongly reduce the amount of data necessary for effective parameter learning.

Section 2 comments on some related work about parameter learning with qualitative knowledge. Section 3 introduces our notation for Influence Diagrams and the problem of probabilistic parameter learning. Section 4 details a collection of qualitative constraints that can guide the learning process. Then we describe a procedure to solve parameter learning by reformulating the problem as a constrained convex optimization problem (Section 5). Section 6 presents some experiments with synthetic data, and finally section 7 concludes the report and indicates future work.

2 Related Work

Domain knowledge can be classified as quantitative and qualitative, which describes the explicit quantification of parameters and approximate characterizations, respectively. Both are useful for parameter learning, but quantitative knowledge has been widely used while qualitative relations among parameters have not been fully exploited. We focus our attention on related work about qualitative relations. Parameter learning in general is a well explored topic and we suggest Jordan's book [13] as a starting point for a broader review.

Concerning the use of qualitative relations, Wittig et al. [27] present a method to integrate qualitative constraints into two learning algorithms (APN and EM) by introducing violation functions as penalty terms to the log likelihood function. They show that domain knowledge in the form of constraints can improve learning accuracy. However, weights for the penalty functions often need be manually tuned, which strongly rely on human knowledge about such weights. Altendorf et al. [1] describe a method to incorporate monotonicity constraints into the learning algorithm. It is based on the assumption that states of discrete random variables can be totally ordered, as we also do. Additionally, it uses penalty functions, which suffer from the same problem as [27]. Feelders and Van der Gaag [10] incorporate some simple inequality constraints in the learning process. They assume that all the variables are binary. Moreover, the constraints used in all above methods [1, 10, 27] are too restrictive, because constraints can not involve parameters among many distributions and parameters can not appear in as many constraints as we desire.

de Campos and Cozman [7] formulate the learning problem as a constrained optimization problem. However, they are restricted to complete datasets and apply non-convex optimization. Niculescu et al. [16, 17] also solve the learning problem by optimization techniques. They derive closed form solutions for the maximum likelihood estimation supposing some predefined types of constraints, much like the work presented here. However, there are two main limitations of their methods: there is no overlaping between parameters of different constraints in general, that is, parameters are restricted to appear in only one constraint at all and constraints are restricted to single distributions. There are very restricted cases where parameters and constraints can involve distinct distributions. We describe a general learning procedure where such limitations do not exist.

Similar ideas are proposed by Xue and Ji [29]. They derive closed form solutions for the maximum likelihood estimation and the EM algorithm supposing some predefined types of constraints. We emphasize that the work presented here is more general, because we allow potentially any convex constraint among parameters to be included in the model and we guarantee convergence to a global

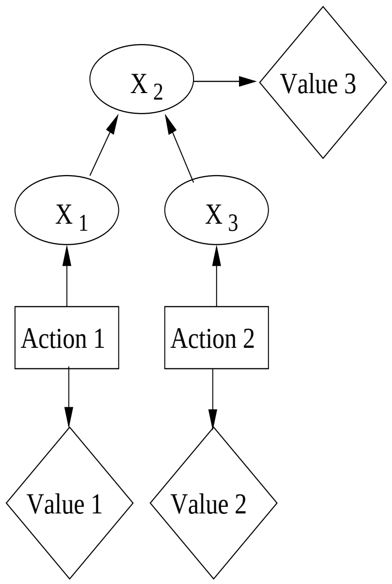


Figure 1: Simple Influence Diagram example.

optimum of the problem. Such properties were not obtained before.

3 Problem definition

An Influence Diagram is a directed acyclic graph where nodes may have three distinct types: chance, decision and value nodes. Links between nodes characterize conditional dependencies among them. Explicitly, links toward a chance node indicate probabilistic dependency of the node on its parents; links to a decision node indicate what information is available to take the decision, and links to value nodes represent a utility for the parents (value nodes cannot have children). Associated to each node, there are parameters:

1. *Chance nodes*: for each such node, there is an associated discrete random variable X_i with domain $\Omega_{X_i} = x_i^1, \ldots, x_i^{r_i}$ and conditional probability distributions $p(X_i|\pi(X_i))$, where $\pi(X_i)$ are the parents of X_i in the graph. In fact there must exist one distribution for each $\pi^j(X_i)$, where j is viewed as a shortcut for a complete configuration of the parents of X_i, that is, $\pi^j(X_i)$ defines a set $(x_{i_1}^{k_1}, x_{i_2}^{k_2}, \ldots, x_{i_t}^{k_t})$, where $t = |\pi(X_i)|$, $X_{i_a} \in \pi(X_i)$ and $k_a \in \{1, \ldots, r_{i_a}\}$ for $a = 1, \ldots, t$. If a variable has no parents, then we define j equals to zero. Whenever necessary and for ease of expose, we use an

extended notation $j \doteq (x_{i_1}^{k_1}, x_{i_2}^{k_2}, \ldots, x_{i_t}^{k_t})$. Therefore, x_i^k represents the kth state of random variable X_i, and $\pi^j(X_i)$ is the jth parent configuration of it. Furthermore, let $\mathcal{X} = \{X_1, \ldots, X_n\}$ be the set of chance nodes/random variables (we use them interchanged if no confusion arises) and q_i the number of distinct configuration of $\pi(X_i)$ (that is, $q_i = \prod_{X_t \in \pi(X_i)} r_t$).

2. *Decision nodes*: for each such node, there is a decision variable Y_i with domain $\Omega_{Y_i} = y_i^1, \ldots, y_i^{s_i}$. Besides that, the configuration of $\pi(Y_i)$ is available at the time of the decision. A *policy* for Y_i, denoted δ_i, specifies the action to take, that is, $\delta_i \in \Omega_{Y_i}$. We denote by $\mathcal{Y}$ the set of decision nodes, and by δ a set of policies δ_i, one for each decision node Y_i (this is also called *strategy*).

3. *Value nodes*: for each such node, there is a function $g_{U_i} : \Omega_{\pi(U_i)} \to \mathcal{Q}$, where $\Omega_{\pi(U_i)}$ is the set of all possible configuration of $\pi(U_i)$ and $\mathcal{Q}$ are the rational numbers. g_{U_i} defines the utility for each configuration of the parents. We define $\mathcal{U}$ as the set of all value nodes.

A simple example is depicted in Figure 1. Decision nodes are represented by rectangles, chance nodes by ellipses and value nodes by diamonds.

Example 1 *Using the Influence Diagram of Figure 1, we can model* Action 1 *as a ground attack to enemy positions,* Action 2 *as bombing such positions,* Value 1 *as the cost of a ground attack, and* Value 2 *as the cost of bombing.* X_2 *is our goal, for instance to achieve territory occupation, while* Value 3 *is the profit of the goal.* X_1 *and* X_3 *represent the uncertain outcome of* Action 1 *and* Action 2, *respectively.*

Given an strategy δ and a complete configuration for the random variables $\mathbf{x} = (x_1^{k_1}, \ldots, x_n^{k_n})$, we can compute the joint probability as follows

$$p_\delta(\mathbf{x}) = \prod_{i=1}^{n} p(x_i^{k_i} | \pi^j(X_i)) \mathcal{I}(x_i^k \cup \pi^j(X_i) \subseteq \mathbf{x} \cup \delta),$$

where $\mathcal{I}$ is an indicator function determining that x_i^k and $\pi^j(X_i)$ must cope with the configuration of $\mathbf{x}$ and δ. If we see δ as evidence, then this formula is just similar to the joint probability of well-known Bayesian networks. Marginal probabilities can be obtained summing over the undesired variables, that is, if $\mathbf{x}' \subset \mathbf{x}$, then

$$p_\delta(\mathbf{x}') = \sum_{x'' \in \Omega_{\mathbf{x} \setminus \mathbf{x}'}} p_\delta(\mathbf{x}).$$

For a given strategy, the main inference in an Influence Diagram is to evaluate its expected utility, which is obtained using a weighted sum of corresponding utility functions:

$$E_\delta = \sum_{U \in \mathcal{U}} \sum_j p_\delta(\pi^j(U)) g_U(\pi^j(U)),$$

where j ranges in all possible parent configurations of U.

Note that inferences in Influence Diagrams rely on the quality of the probabilistic parameters to weight the utility functions. This report focuses on learning of such parameters. Hence, the graph structure and the utility functions are supposed to be known in advance. To ease of expose, we define θ as the entire vector of probabilistic parameters, where $\theta_{ijk} = p(x_i^k | \pi^j(X_i))$. Moreover, we need to specify an order for the states of each random variable, that is, for each X_i, we define an order $x_i^{k_1} < x_i^{k_2} < \ldots < x_i^{k_{r_i}}$, where $\{k_1, \ldots, k_{r_i}\} \subseteq \{1, \ldots, r_i\}$. Without loss of generality, we suppose that $k_s = s$ for all $s \in \{1, \ldots, r_i\}$ (if necessary, we could exchange the position of some states to comply with this rule). Thus we have $x_i^1 < x_i^2 < \ldots < x_i^{r_i}$ for all X_i.

3.1 Learning probabilistic parameters

Given a dataset $D = \{D_1, \ldots, D_N\}$, with $D_t = (x_{1,t}^{k_1}, \ldots, x_{n,t}^{k_n}, \delta)$ a sample of the chance and decision nodes, the goal of parameter learning is to find the most probable values for θ. These values best explain the dataset D, which can be quantified by the log likelihood function $\log(p(D|\theta))$, denoted $L_D(\theta)$. Assuming that samples are drawn independently from the underlying distribution, we have

$$L_D(\theta) = \log \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}}, \tag{1}$$

where n_{ijk} indicates how many elements of D contain both x_i^k and $\pi^j(X_i)$.

If the dataset D is complete, Maximum Likelihood (ML) estimation method can be described as a constrained optimization problem, i.e. maximize Equation (1), subject to simplex equality constrains:

$$\begin{array}{lll} \max & & L_D(\theta) \\ s.t. & \forall_{i=1,\ldots n} \forall_{j=1\ldots q_i} & g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0 \end{array} \tag{2}$$

where $g_{ij}(\theta) = 0$ imposes that distributions defined for each variable given a parent configuration sums one over all variable states. This problem has its global optimum solution at $\theta_{ijk} = \frac{n_{ijk}}{n_{ij}}$, where $n_{ij} = \sum_{k=1,\ldots,r_i} n_{ijk}$.

4 Qualitative Constraints

Standard likelihood estimations are usually enough if we have enough data, regardless one uses prior knowledge or not. However, when small amount of data is available, the likelihood function may not produce reliable estimations for the parameters.

Example 2 *Suppose we are working with the chance nodes X_1, X_2, X_3 of Example 1, and they are associated to binary random variables (with categories x_i^1 meaning true and x_i^2 meaning false). Suppose further that we have the dataset $D = (D_1, D_2)$, with $D_1 = (x_1^1, x_2^1, x_3^1)$ and $D_2 = (x_1^2, x_2^2, x_3^2)$. Using the ML estimation, we have the posterior probabilities $\theta_{101} = \theta_{102} = \theta_{301} = \theta_{302} = 0.5$ and $\theta_{2j_11} = \theta_{2j_22} = 1$, with $j_1 \doteq (x_1^1, x_3^1)$, $j_2 \doteq (x_1^2, x_3^2)$, $j_3 \doteq (x_1^2, x_3^1)$, $j_4 \doteq (x_1^1, x_3^2)$. Posterior probability distributions θ_{2j_3k} and θ_{2j_4k} can not be estimated as no data about such configurations are available.*

Situations like in Example 2 could be alleviated by inserting quantitative prior distributions for the parameters. However, acquiring such quantitative prior information may not be an easy task. An incorrect quantitative prior might lead to bad estimation results. For example, standard methods apply quantitative uniform priors. In this case, if no data are present for a given parameter, then the answer would be 0.5, which may be far from the correct value.

A path to overcome this situation is through qualitative information. Qualitative knowledge is likely to be available even when quantitative knowledge is not, and tends to be more reliable. For example, someone hardly will make a mistake about the qualitative relation between sizes of the Earth and the Sun; almost everyone will fail to specify a quantitative ratio (even approximate).

Example 3 *Suppose, in addition to Example 2, that the following two constraints are known: $\theta_{302} + \theta_{2j_31} \leq 0.7$ and $\theta_{2j_11} \leq \theta_{2j_42}$. With this knowledge, we can state that $\theta_{2j_31} \leq 0.2$ and $\theta_{2j_42} = 1$, reducing the space of possible parameters and alleviating the problem with sparse quantitative data.*

We start by describing constraints that appear in Qualitative Probabilistic Networks [25]. The most common type of qualitative constraints is called *influence*. Qualitative influences define some knowledge about the state of a variable given the state of another. For example, the probability of achieving territory occupation given that a successful ground attack was employed is greater than when it was not.

Definition 1 *Let X_a, X_b be variables such that $X_a \in \pi(X_b)$. We say that X_a influences X_b positively if*

$$\forall_{k_a > k'_a, k_b = 1,\ldots,r_b - 1} \sum_{k > k_b} \theta_{bj_{k_a}k} \geq \sum_{k > k_b} \theta_{bj_{k'_a}k} + \delta, \tag{3}$$

where $j_k \doteq (x_a^k, \pi^{j_*}(X_b))$ *and* j_* *is an index ranging over all parent configurations except for* X_a*, that is,* $\pi^{j_*}(X_b) = (x_i^{k_i} : X_i \in \pi(X_b) \wedge i \neq a \wedge k_i \in \{1, \ldots, r_i\})$*, and* $\delta \geq 0$ *is a constant.*

This roughly means that observing a greater state for X_a makes more likely to have greater states in X_b. The definition makes use of cumulative probability values, so it works even for non-binary variables. When applied to binary variables, summations of Equation (3) disappear and we have a more natural formulation: $\theta_{bj_2 2} \geq \theta_{bj_1 2} + \delta$, with j_k as in Definition 1. In this case, the greater state is 2, and observing x_a^2 makes more likely to have x_b^2.

If constraints of Definition 1 hold for $\delta > 0$, the influence is said *strong* with threshold δ [20]. Otherwise, it is said *weak* for δ. A negative influence is obtained by replacing the inequality operator $\geq$ by $\leq$ and the sign of the δ term to negative in Equation (3). A zero influence is obtained by changing inequality to an equality.

We now define a conjugate influence from two parents to a common child, called additive synergy [25]. Synergies are influences from two parents acting to influence the child. For example, suppose we bomb an area and/or perform a ground attack. Both actions contribute to the goal. But the probability of success given that we do both or neither is lesser than the probability of success given that we do only one of them (bombing an area under ground attack could kill our own infantry). So, the actions together have a negative influence on achieving the goal.

Definition 2 *Let* X_a *and* X_c *be parents of* X_b*. We say that* X_a *and* X_c *impose a negative additive synergy on* X_b *if*

$$\forall_{k_b=1,\ldots,r_b-1} \\ \sum_{k>k_b} \sum_{k_a=k_c} \theta_{bj_{k_a,k_c}k} \leq \sum_{k>k_b} \sum_{k_a \neq k_c} \theta_{bj_{k_a,k_c}k} - \delta \tag{4}$$

where $j_{k_a,k_c} \doteq (x_a^{k_a}, x_c^{k_c}, \pi^{j_*}(X_b))$ *and* j_* *ranges over all parent configurations not including* X_a *nor* X_c*, and* $\delta \geq 0$ *is a constant.*

This means that observing the same configuration for the parents X_a and X_c makes less likely to have a greater state in X_b. Again the case over binary variables is simpler: $\theta_{bj_{1,1}2} + \theta_{bj_{2,2}2} \leq \theta_{bj_{1,2}2} + \theta_{bj_{2,1}2} - \delta$, where j_{k_a,k_c} is as in Definition 2, which enforces the sum of parameters with equal configurations for X_a and X_c to be lesser than the sum of parameters with distinct configurations. If these constraints hold for $\delta > 0$, this synergy is said *strong* with threshold δ [20]. Otherwise it is said *weak* for δ. When omitted, δ is assumed to be zero. Positive and zero additive synergies are obtained analogously.

Non-monotonic influences and synergies happen when constraints hold only for some configurations of inactive parents (regarding that constraint) [21]. For example, suppose three binary variables such that X_1 has X_2 and X_3 as parents and that $\theta_{1(x_2^2,x_3^1)2} \geq \theta_{1(x_2^1,x_3^1)2}$ holds, but $\theta_{1(x_2^2,x_3^2)2} \geq \theta_{1(x_2^1,x_3^2)2}$ can not be stated. Hence we do not have a positive influence of X_2 on X_1, because it would be necessary to have both constraints valid to ensure that influence. In fact we might realize that the state of X_3 (the inactive parent concerning this influence) is important in the constraint. Then, we may state a non-monotonic influence of X_2 on X_1 that holds when X_3 is x_3^1 but not when it is x_3^2. Situational signs [4] and context-specific signs [22] are special cases of such non-monotonic constraints. To include all such situations, we define a very general constraint: a *linear relationship constraint* defines a linear relative relationship between sets of parameters and numerical bounds.

Definition 3 *Let θ_A be a sequence of parameters, α_A a corresponding sequence of constant numbers and α also a constant. A* linear relationship constraint *is defined as*

$$\sum_{\theta_{ijk} \in \theta_A} \alpha_{ijk} \cdot \theta_{ijk} \leq \alpha, \tag{5}$$

that is, any linear constraint over parameters can be expressed as a *linear relationship constraint*. It is worth to mention some special cases: if θ_A has only one parameter θ_{ijk} and $\alpha_{ijk} = 1$, the constraint becomes a upper bound constraint for θ_{ijk} (we can obtain a lower bound using negative α_{ijk} and α). These bounds are also known as *range constraints* [16]. If all parameters involved in a linear relationship constraint share the same node index i and parent configuration j, the constraint is called *intra-relationship constraint*. Otherwise, it is a *inter-relationship constraint*. We indicate this difference because usually *inter-relationship constraints* lead to hard learning procedures [16]. Here, as we will discuss, there is no important distinction between them regarding complexity. We note that *linear relationship constraints* generalize influences and additive synergies and corresponding non-monotonic relations, as all of them are linear constraints over parameters. However we decided to keep such definitions separately given their importance in the literature.

5 Learning through convex optimization

The definitions of previous section show many types of qualitative constraints that can be used to describe our knowledge. In this section we present an optimization idea to solve the learning problem using a great variety of constraints. The main achievements are

- It is possible to mix different types of constraints in a straightforward way.

- Parameters of the network can appear as many times as desired.
- There is no distinction about creating a qualitative constraint within a single probability distribution or among parameters in many distributions.
- Many types of qualitative relations can be handled (any convex constraint), including usual relations defined in the literature [4, 16, 17, 18, 22, 25].

We describe how constraints and the log likelihood function can be formulated as a convex constraint of a convex optimization program. Such a program can be specified as [3]:

$$\begin{aligned} \min_{\theta} \quad & f(\theta) \\ s.t. \ \forall_{i=1,\dots,m} \quad & g_i(\theta) \quad \leq 0 \end{aligned} \tag{6}$$

where θ is the vector of optimization variables, m is the number of constraints and f and g_i (for all i) are continuous convex functions over the parameter space.

First note that our objective function is concave because $\log$ is concave and a positive linear combination of concave functions is also concave (each n_{ijk} is positive, known from the dataset. If $n_{ijk} = 0$, the term is simply discarded). Thus

$$\max_{\theta} \left(\log \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \right) = \min_{\theta} - \sum_{i,j,k} n_{ijk} \cdot \log \theta_{ijk},$$

where the right-hand side is a minimization of a convex function (-f is convex when f is concave), as required in Equation (6). Regarding constraints, any linear function is convex. Simple manipulations can lead them to the form of Equation (6) (a constraint $h(\theta) \geq 0$ just need to be multiplied by -1, while an equality can be viewed as two inequalities). All constraints defined in Section 4 are convex and can be directly inserted into the convex program.

To solve convex programming, there are many optimization algorithms. We can use specialized interior point solvers [2] or even some general optimization ideas [15], because convex programming has the attractive property that any local optimum is also a global optimum. Furthermore, such global optimum can be found in polynomial time in the size of input [3].

5.1 Incomplete data

Incomplete data means that some fields of the dataset are unknown. If the dataset is $D = \{D_1, \dots, D_N\}$, then each $D_t \subseteq (x_{1,t}^{k_1}, \dots, x_{n,t}^{k_n})$ is a sample of some nodes. We say that u_t is the missing part in tuple t, that is, $u_t \cap D_t = \emptyset$ and $u_t \cup D_t$ is a complete instantiation for the nodes. Let U be the set of all missing data. In this case, the likelihood function $\log(p(D|\theta))$ is not a simple product anymore, and the corresponding optimization program is not convex.

A common method to overcome this situation is standard EM algorithm [8], which starts from some initial guess, and then iteratively takes two types of steps (E-steps and M-steps) to get a local maximum of the likelihood function. Particularly for discrete nodes, E-step computes the expected counts for all parameters, and M-step estimates the parameters by maximizing log likelihood function, given the counts from E-step, just like would be done with a complete dataset. EM algorithm converges to a local maximum under very few assumptions [28].

Assume θ^0 is an initial guess for the parameters, and θ^t denotes the estimation after t iterations, $t = 1, 2, \ldots$. Then, each iteration of EM can be summarized as follows:

- *E-step:* compute expectation of the log likelihood given observed data D and current estimation of parameters θ^t: $Q(\theta|\theta^t) = E_{\theta^t}[\log p(U \cup D|\theta)|\theta^t, D]$.
- *M-step:* find new parameter θ^{t+1}, which maximizes expected log likelihood computed in E-step: $\theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t)$.

We propose to extend EM with the convex optimization program of Section 5, that is, the M-step is performed using convex programming. The value of θ^{t+1} is $\arg\max_{\theta} Q(\theta|\theta^t)$ subject to qualitative constraints of the problem, and a polynomial time algorithm can be employed to solve this convex program (also described in Section 5). In this context, qualitative constraints may help EM to avoid poor local maxima and improve the overall solution. Furthermore, because the parameter space is convex and the enhanced M-step produces a global optimum solution for the current parameter counts, this modified EM shares convergence and optimality properties of the standard EM algorithm [5, 28]. Although the modified EM is more time expensive than the standard EM as each M-step requires the solution of a convex optimization program (standard EM may use closed form solution for ML), we argue that, just as in standard EM where an improving solution is enough instead of an optimum one (called *Generalized EM*), we might stop the convex programming as soon as an improving solution is found.

6 Experiments

In order to test the performance of our method against standard ML estimation and standard EM algorithm given sparse and incomplete data, we use a similar Influence Diagram as described by Zhang and Ji [30] for an hypothetical EBO-based

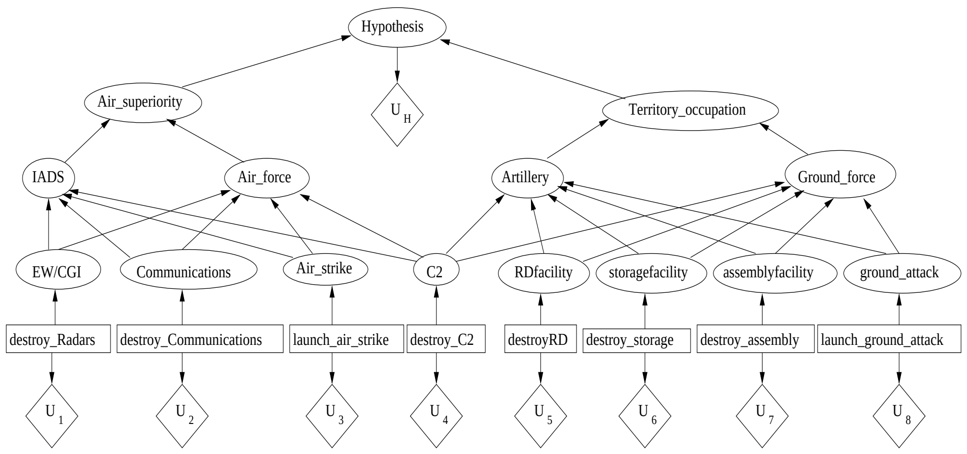


Figure 2: Influence Diagram for an hypothetical EBO-based planning problem.

military planning problem. It is shown in Figure 2. The goal is to win a war and it is represented by the *Hypothesis* node (on top of Figure 2). Just below there are the subgoals *Air_superiority* and *Territory_occupation*, which are directly related to the main goal. There are eight decision nodes (represented by rectangles, while chance nodes are ellipses): *destroy_C2* (C2 stands for *Command and Control*). *destroy_Radars*, *destroy_Communications*, *launch_air_strike*, *destroy_RD*, *destroy_storage*, *destroy_assembly*, *launch_ground_ attack*. Just above decision nodes, we have chance nodes representing the outcomes of performing such actions (they indicate the workability of such systems), and below we have value nodes (diamond-shaped nodes) describing the cost of each action. Furthermore, we have four chance nodes (in the center of the figure) indicating general workability of *IADS* (Integrated Air Defense System), *Air_force*, *Artillery* and *Ground_force* of the enemy. The overall profit of winning is given by the value node U_H, child of *Hypothesis*.

As this is an hypothetical example, we define utility functions and probability distributions as follows:

- Probability of *Hypothesis* is one given that all subgoals are achieved. If one of subgoals is not achieved, then the probability of *Hypothesis* is 60%, and if none of subgoals is achieved, then we certainly fail in the campaign.
- For the subgoals *Air_superiority* and *Territory_occupation*, we define them as accomplished with probability one when both children were achieved, 50% when only one child is achieved, and zero when none is achieved.
- For the probabilities of *IADS*, *Air_force*, *Artillery* and *Ground_force*, we de-

fine a decrease of 30% for each unaccomplished child (with a minimum of zero, of course). For instance, *IADS* has probability 40% if two of its four children are achieved, and *Artillery* has probability zero when at least four of its children are unaccomplished.

- The probability of the outcomes of actions (chance nodes just above decision nodes), we define a rate of 90% of success. For example, the decision *destroy_Radars* will have *EW/GCI_radars* destroyed with 90% of odds (EW/GCI means *Early Warning/Ground Control Interception*).
- The reward of achieving the main goal is 1000, while not achieving it costs 500.
- Costs of actions are as follows: *ground_attack* is 150, *air_strike* is 50, and other actions cost 20 each.

Using this Influence Diagram as our "truth" for the parametrization, we generate samples from it. After training the model with distinct amounts of data, we apply the Kullback-Leibler (KL) divergence criterion [14] to measure the difference between joint probability distributions induced by generated diagrams and distributions of the "truth" diagram. For probability distributions P_1 and P_2 over discrete variables, where P_1 is considered the "truth", the KL divergence is

$$KL(P_1, P_2) = \sum_i P_1(i) \log \frac{P_1(i)}{P_2(i)},$$

where i ranges over all possible configurations of the discrete variables and $P_j(i)$ means the probability value for the configuration i. We have chosen to use such criterion because the problem is hypothetical and no real data is available. Thus, it would not be significant to compare maximum expected utilities of the diagrams generated with each approach, because we could obtain a great expected utility even using wrong parameters. Such result would be unreliable, as we desire to obtain parameters that best describe a real situation, and those parameters could not lead to good results. So, using the KL divergence, we measure how much improvement the qualitative knowledge obtains for parameter learning towards the correct values.

We conduct experiments for datasets with 10, 100 and 1000 samples, without and with qualitative constraints. When employed, the qualitative constraints state that each chance or decision node has a positive influence on its children, that is, achieving the parent makes more likely to accomplish the children. We use the following constraints. Although they are hypothetical, we tried to create as meaningful constraints as possible.

- *Air_superiority* has a strong positive influence on *Hypothesis*, as well as *Territory_occupation*, which means that the probability of success on *Hypothesis* given that we have achieved *Air_superiority* is much greater than the the probability of success on *Hypothesis* given that we have not achieved *Air_superiority*, as subgoals contribute to the main goal. Analogously, the probability of *Hypothesis* given we have success on *Territory_occupation* is much greater than the probability of *Hypothesis* given failure on *Territory_occupation*.

- *IADS* and *Air_force* have strong positive influences on *Air_superiority*, which means that the probability of *Air_superiority* given each one of those achievements is much greater than when we do not obtain them. Again, *IADS* and *Air_force* are important steps to achieve *Air_superiority*, so they contribute positively to it.

- *Artillery* and *Ground_force* have positive influence on *Territory_occupation*, so the probability of the latter is greater when *Artillery* and/or *Ground_force* are accomplished.

- Destroyed *EW/GCI radars* has negative influence on *IADS* and *Air_force* of the enemy, as well as inoperable *Communications* and *C2* and performed *Air_strike*.

- Destroyed *RDfacility* has negative influence on *Artillery* and *Ground_force* of the enemy, as well as inoperable *C2*, *storagefacility*, *assemblyfacility* and performed *ground_attack*.

- Probability of *Air_superiority* given success on *IADS* and *Air_force* is greater than the probability of *Territory_occupation* given *Artillery* and *Ground_force*, because we consider (in this example) that it is easier to control the airspace than to control the surface.

For each amount of data (10, 100 and 1000 samples), we work with 20 random sets of data and qualitative constraints. Mean and variance are presented in first three lines of Table 1 for complete data. The last three lines of the table show results for datasets with missing fields (chosen at random in number equal to three times the number of samples). In such cases, EM and constrained EM methods are employed.

Table 1 indicates a decrease in the divergence when working with qualitative constraints, which shows that such constraints were actively used during the learning process. For sparse data, the divergence was substantially reduced so as harder

Amount of data	Mean		Variance	
	Unconstrained	Constrained	Unconstrained	Constrained
10	8.4	2.39	0.35	0.64
100	2.25	0.32	0.09	0.03
1000	0.06	0.08	0.0005	0.00004
10 (I)	8.12	2.46	0.15	0.1
100 (I)	2.33	0.36	0.15	0.02
1000 (I)	0.08	0.08	0.0004	0.0001

Table 1: KL divergence for 20 runs of the learning procedure using random samples and constraints. Unconstrained results are the standard case (ML / EM ideas) while Constrained indicate the use of qualitative relations during learning (Constrained ML / Constrained EM). Rows marked with (I) were executed with incomplete datasets.

problems are most benefited. Besides that, results show a somewhat expected situation: when enough data is available, qualitative constraints become less useful. We can even see a case where the average result with qualitative constraints was sightly worse than without constraints (1000 samples without missing data). That situation eventually happens when enough data is available and there exist qualitative constraints that are not precise with respect to the ground truth. Thus, we emphasize that problems like EBO-based planning may have strong benefits from qualitative knowledge, as usually there are not enough data, while problems where enough data are available may not have the same benefits.

Figures 3 and 4 show the difference in the KL divergence between the unconstrained learning (standard case) and the constrained learning with qualitative relations for each node of the Influence Diagram. Black bars are for 10-sample cases and white bars for 100-sample cases. Node numbers in these graphs are defined from 1 to 23 according to the following order: (1) *C2*, (2) *EW/GCI*, (3) *Communications*, (4) *Air_strike*, (5) *IADS*, (6) *Air_force*, (7) *Air_superiority*, (8) *RDfacility*, (9) *storagefacility*, (10) *assemblyfacility*, (11) *ground_attack*, (12) *Artillery*, (13) *Ground_force*, (14) *Territory_occupation*, (15) *Hypothesis*. We can see that most benefits of qualitative constraints appear in nodes that are hard to learn, for instance *Hypothesis*, its two children *Air_superiority* and *Territory_occupation*, and their children. We can verify that advantages with complete and incomplete datasets were very similar to each other, which shows that even with incomplete data the qualitative constraints help to guide the learning process.

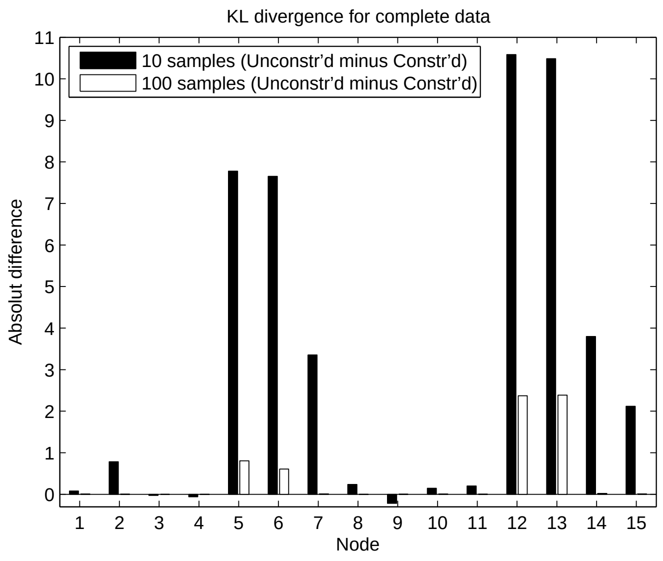


Figure 3: Difference in KL divergence between unconstrained (standard) learning and constrained learning for complete data. Positive bars mean that the constrained version performed better than the unconstrained one.

7 Conclusion

This report describes a framework for parameter learning of Influence Diagrams when qualitative knowledge is available, which is specially important for training with sparse data. Even with enough data, qualitative constraints may help to guide the learning procedures.

For complete data, we directly apply convex optimization to obtain a global optimum of the constrained maximum likelihood estimation, while for incomplete data, we extend the EM method by introducing a constrained maximization in the M-step. We apply our method to synthetic Influence Diagrams based on EBO military planning. Experimental results demonstrate that our method can fully exploit the qualitative knowledge to improve parameter learning accuracy. With sparse data and constraints, it is possible to obtain results similar to those of using expressively more data without constraints.

For future research, we intend to apply the ideas on harder planning problems where only sparse data is available. We plan to explore other applications of qualitative constraints, such as strategy evaluation (trying to reduce the time for each evaluation) and planning (constraints may guide the search for the best strategy). Furthermore, qualitative constrains presented here can be viewed as hard

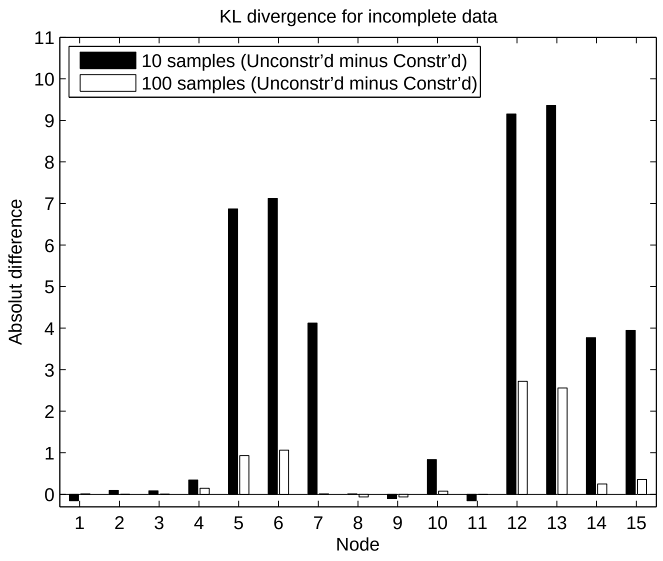


Figure 4: Difference in KL divergence between unconstrained (standard) learning and constrained learning for incomplete data. Positive bars mean that the constrained version performed better than the unconstrained one.

constraints, since they must be satisfied during the learning. We intend to explore, together with hard constraints, some soft qualitative constrains in the sense that estimations may eventually not comply with them (to avoid possible "imprecise" constraints).

References

[1] E. Altendorf, A. C. Restificar, and T. G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *UAI*, pages 18–26, 2005. 3

[2] E. D. Andersen, B. Jensen, R. Sandvik, and U. Worsoe. The improvements in mosek version 5. Technical report, Mosek Aps, 2007. 10

[3] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. MPS/SIAM Series on Optimization. SIAM, 2001. 10

[4] J. H. Bolt, L. C. van der Gaag, and S. Renooij. Introducing situational influences in QPNs. In *ECSQARU*, pages 113–124, 2003. 2, 9, 10

[5] R. A. Boyles. On the convergence of the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 45(1):47–50, 1983. 11

[6] P. Davis. Effects-based operations: a grand challenge for the analytical community. Technical report, Rand corp., 2003. MR1477. 1

[7] C. P. de Campos and F. G. Cozman. Belief updating and learning in semi-qualitative probabilistic networks. In *UAI*, pages 153–160, 2005. 3

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 11

[9] D. A. Deptula. Effects-based operations: change in the nature of warfare. *Defense and Airpower Series*, pages 3–6, 2001. 1

[10] A. Feelders and L. C. van der Gaag. Learning Bayesian network parameters under order constraints. *International Journal of Approximate Reasoning*, 42(1-2):37–53, 2006. 3

[11] R. A. Howard and J. E. Matheson. *Influence diagrams*, volume II. Strategic Decisions Group, Menlo Park, CA, 1984. 1

[12] Qiang Ji. Modeling and evaluating ebo-based miltiary planning using the influence diagram. Technical report, Rensselaer Polytechnic Institute, Troy, NY, 2007. ARO TR 2007-01. 2

[13] M. Jordan, editor. *Learning Graphical Models*. The MIT Press, 1998. 3

[14] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. on Information Theory*, 37(1):145–151, 1991. 13

[15] B. A. Murtagh and M. A. Saunders. Minos 5.4 user's guide. Technical report, Systems Optimization Laboratory, Stanford University, 1995. 10

[16] R. S. Niculescu. *Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks*. PhD thesis, Carnegie Mellon, 2005. CMU-CS-05-147. 2, 3, 9, 10

[17] R. S. Niculescu, T. Mitchell, and B. Rao. Bayesian network learning with parameter constraints. *Journal of Machine Learning Research*, 7(Jul):1357–1383, 2006. 3, 10

[18] R. S. Niculescu, T. M. Mitchell, and R. B. Rao. A theoretical framework for learning bayesian networks with parameter inequality constraints. In *IJCAI*, pages 155–160, 2007. 10

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988. 1

[20] S. Renooij and L. C. van der Gaag. Enhancing QPNs for trade-off resolution. In *UAI*, pages 559–566, 1999. 2, 8

[21] S. Renooij and L. C. van der Gaag. Exploiting non-monotonic influences in qualitative belief networks. In *IPMU*, pages 1285–1290, Madrid, Spain, 2000. 8

[22] S. Renooij, L. C. van der Gaag, and S. Parsons. Context-specific sign-propagation in qualitative probabilistic networks. *Artificial Intelligence*, 140(1-2):207–230, 2002. 9, 10

[23] R. D. Shachter. Evaluating influence diagrams. *Operations Research*, 34:871–882, 1986. 1

[24] L. C. van der Gaag, H. L. Bodlaender, and A. Feelders. Monotonicity in Bayesian networks. In *UAI*, pages 569–576. AUAI Press, 2004. 2

[25] M. P. Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial Intelligence*, 44(3):257–303, 1990. 2, 7, 8, 10

[26] M. P. Wellman and M. Henrion. Explaining 'explaining away' . *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):287–307, 1993. 2

[27] F. Wittig and A. Jameson. Exploiting qualitative knowledge in the learning of conditional probabilities of Bayesian networks. In *UAI*, pages 644–652, 2000. 3

[28] C. F. Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, Mar 1983. 11

[29] Zheng Xue and Qiang Ji. Exploiting qualitative constraints for learning bayesian networks under insufficient data. Technical report, Rensselaer Polytechnic Institute, Troy, NY, 2007. 3

[30] W. Zhang and Q. Ji. A factorization approach to evaluating simultaneous influence diagrams. *IEEE Transactions on Systems, Man and Cybernetics A*, 36(4):746–757, 2006. 11

Exploiting Qualitative Constraints for Learning Bayesian Networks under Insufficient Data

Zheng Xue and Qiang Ji
Rensselaer Polytechnic Institute

Abstract

Graphical models (GMs) such as Bayesian Networks (BN) or the Influence Diagrams (ID) are being increasingly applied to many different applications. One bottleneck in using GMs is that learning the GM model parameters often requires a relative large amount of training data. However, in real life and for many applications, training data is often incomplete or sparse, which can cause low learning accuracy. Incorporating domain knowledge can help alleviate this problem. Instead of using quantitative prior knowledge as used by most of the existing methods, this paper introduces a novel learning method based on systematically combining the training data with some qualitative knowledge.

To validate our method, we compare it with the Maximum Likelihood (ML) estimation method under sparse data and with the Expectation Maximization (EM) algorithm under incomplete data respectively. The experimental results show that our method improves the parameter learning accuracy significantly compared with both ML and EM algorithms.

1 Introduction

Among all the issues of graphical models, parameter learning is one of the main challenges. Parameter learning is to estimate the entries of the conditional probability distributions (CPDs) given the structure of a model. Many learning techniques rely heavily on training data [7]. Ideally, with sufficient data, it is possible to learn the parameters by standard statistical analysis like maximum likelihood (ML) estimation. In many real-world cases, however, the data are either incomplete or sparse, which can cause inaccurate parameter estimation. Data incompleteness is defined as missing of data for some parameters, while data sparseness means the amount of training data is limited.

When data are incomplete, Expectation-Maximization (EM) [3] algorithm is often used. Most EM-based methods work under the assumption that data are missing at random (MAR), which means the missing values can be estimated by the observed ones in some way. However, when data are missing completely at random (MCAR), e.g data of hidden nodes, the learned parameters could be far from the ground truth. The reason is that the missing data do not even depend on the observed ones, and there is no way to estimate the missing data only from the observed ones.

In our paper, we propose a framework to solve the parameter learning problem by combining quantitative data and domain knowledge in the form of qualitative constraints. Two kinds of qualitative constraints are defined: range constraints which are applied to individual parameters; and relationship constraints which are applied to pairs of parameters. For sparse but complete data, we solve the learning task by reformulating the problem as a constrained ML (CML) problem. For incomplete data, we introduce the constrained EM (CEM) by adding constraints to the M step, and iteratively solve the learning problem. In addition, we provide closed form solutions to both CML and CEM.

2 Related Work

We have already discussed that one of the shortcomings of EM algorithms is that it can easily be trapped in a local maximum when data are MCAR. Till now, there are many different methods to help EM to escape from the local maximum, such as the information-bottleneck EM algorithm [4], data perturbing method [5], and AI&M procedure [8]. These methods focus on improving the machine learning techniques, but ignoring the useful domain knowledge.

Domain knowledge can be classified as quantitative and qualitative knowledge, which describe the explicit quantification of parameters, and approximate characterizations of parameters respectively. Both kinds of domain knowledge are useful for parameter learning. While the quantitative knowledge has been widely used in the form of prior probability distributions, qualitative constraints have not been fully exploited in parameter learning yet.

Wittig et al. [12] present a method to integrate qualitative constraints into two learning algorithms, APN [9] and EM, by adding violation functions as a penalty term to the log likelihood function. They show that domain knowledge in the form of constraints can improve learning accuracy. However, this penalty-based method cannot guarantee to find the global maximum. Besides, the weights for the penalty functions often need be manually tuned, depending on applications. Altendorf et al. [1] describes a method to incorporate monotonicity constraints into learning algorithm. It is based on the assumption that the values of the variables can be totally ordered. Additionally, it also uses the penalty functions, which suffers from the same problem as [12]. Feelders and Van der Gaag [6] incorporate some simple inequality constraints in the learning process. They assume that all the variables are binary. The constraints used in the above methods [1, 12, 6] are restrictive, as each constraint has to involve all parameters in a conditional probability table (CPT).

Campos and Cozman [2] formulate the learning problem as a constrained optimization problem. However, they do not provide a specific method to solve the optimization problem. Niculescu et al. [11] also solve the learning problem by optimization techniques. They derive the closed form solutions with ML estimation for two kinds of constraints: inequalities between sums of parameters and upper bounds on sum of parameters within a CPT. There are two main limitations of their method: First, they assume one parameter can and only can have one constraint, and there is no overlap between parameters of different constraints. Second, their method cannot handle constraints from different CPTs. We improve their method by deriving the closed form solution for range constraints, which contain both upper bound and lower bound constraints for the same parameters. In addition, the relationship constraints defined in our paper can either be within or between CPTs.

3 Problem Definition and Approach

3.1 Basic Parameter Learning Theory

We focus on parameter learning in a Bayesian Networks with all discrete nodes, where the structure is known in advance. The method can be extended to other graphical models including the IDs. The notations are defined as follows. Assume a BN with n nodes, θ is the entire vector of parameters, and θ_{ijk} denotes one of the parameters. $\theta_{ijk} = p(x_i^k|pa_i^j)$, where i $(i = 1, ..., n)$ ranges over all the variables in the BN, j $(j = 1, ..., q_i)$ ranges over all the possible parent configurations of node (variable) X_i, and k $(k = 1, ..., r_i)$ ranges over all the possible states of X_i. Therefore, x_i^k represents the kth state of node X_i, and pa_i^j is the jth parent configuration of node X_i.

Given a dataset $D = \{D_1, ..., D_N\}$, which consists of samples of the BN nodes, the goal of parameter learning is to find the most probable values $\hat{\theta}$ for θ that can best explain the dataset D, which is usually quantified by the log likelihood function $\log(p(D|\theta))$, denoted as $L_D(\theta)$. Assuming that the examples are drawn independently from the underlying distribution, based on the conditional independence assumptions in BNs, we have the log likelihood function in Eq.(1), where n_{ijk} is the count for the case that node i has the state k, with the state configuration j for its parent nodes.

$$L_D(\theta) = \log \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \tag{1}$$

If the dataset D is complete, ML estimation method can be described as a constrained optimization problem, i.e. maximize (Eq.(2)), subject to n equality constrains (Eq.(3)).

$$\begin{array}{lr} \text{Max} \quad L_D(\theta) & (2) \\ \text{S.T.} \quad g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0 & (3) \end{array}$$

where g_{ij} imposes the constraint that each parameter sums to 1 over all its state, $1 \leq i \leq n$ and $1 \leq j \leq q_i$.

If dataset D is incomplete, ML estimation cannot be applied directly. A common method is standard EM algorithm [3], which starts from some initial point, and then iteratively takes E step and M step to get a local maximum of the likelihood function. Particularly for discrete nodes, E step computes the expected counts for all parameters, and M step estimates the parameters by maximizing log likelihood function, given the counts from E step. EM algorithm can guarantee to converge to a local maximum. However, depending on different initializations, it may converge to different local maxima. When there are a large number of missing data, which means there are many local maxima, EM algorithm can get stuck in a local maximum far away from the global one.

3.2 Qualitative Constraints

We introduce two kinds of qualitative constraints, which can be easily specified by domain experts. They are range and relationship constraints.

Range constraint defines the upper bound and lower bound of some parameters. Assuming α_{ijk} and β_{ijk} are the upper bound and lower bound for parameter θ_{ijk}, then the range constraints can be defined as follows:

$$\beta_{ijk} \leq \theta_{ijk} \leq \alpha_{ijk} \qquad (4)$$
$$\text{where} \quad 0 < \alpha_{ijk} \leq 1 \text{ and } 0 \leq \beta_{ijk} < 1$$

Relationship constraint defines the relative relationship between a pair of parameters. If both of the two parameters in a relationship constraint share the same node index i, and parent configuration j, the constraint is called *intra-relationship constraint*, which can be represented as follows:

$$\theta_{ijk} \leq \theta_{ijk'} \text{ where } k \neq k' \qquad (5)$$

If the two parameters in a relative relationship constraint do not satisfy the requirement of an intra-relationship constraint, the constraint is called *inter-relationship constraint*. It can be described as follows:

$$\theta_{ijk} \leq \theta_{i'j'k'} \text{ where } i \neq i' \text{ or } j \neq j' \qquad (6)$$

3.3 Overview of Our Approach

We aim to solve the learning problem by reformulating the problem as a constrained based optimization problem, i.e.,

$$\begin{array}{llll} \text{Max} & L_D(\theta) & & (7) \\ \text{S.T.} & g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0, & 1 \leq i \leq n, \text{ and } 1 \leq j \leq q_i & \\ & h_p(\theta) \leq 0, & 1 \leq p \leq S & \end{array}$$

where $h_p(x) \leq 0$ denotes the inequality constraints, and S is the total number of inequality constraints. Using the Lagrange multipliers λ_{ij} and μ_p, the objective function to be maximized can be incorporated with the constraints, producing the following augmented objective function

$$f(\theta) = L_D(\theta) - \sum_{i=1}^{n} \sum_{j=1}^{q_i} \lambda_{ij} g_{ij}(\theta) - \sum_{k=1}^{S} \mu_p h_p(\theta) \qquad (8)$$

Given Eq.(8), for sparse but complete data, we can directly apply the CML method by maximizing Eq.(8) to estimate the parameters. For incomplete data, we can replace the M step of EM algorithm by the solution to Eq.(8), and iteratively obtain the estimation of the parameters. In the section to follow, we introduce our solution to Eq.(8).

4 Parameter Learning With Qualitative Constraints

In this section, we derive the closed form solutions for maximizing Eq.(8) under different types of constraints. Because of the decomposability of the log likelihood function, we can deal with small independent optimization subproblems on independent parameter sets separately instead of dealing with all parameters simultaneously. For this, we define two kinds of parameter sets: one is the *baseline set*, which contains parameters with the same node and the same parent configuration; the other is the *combined set*, which contains several baseline sets. We first separate parameters into baseline sets, and then if there is a constraint on parameters from different baseline sets, we combine those baseline sets into one new combined set. This process continues until there is no constraint on parameters from different sets. After decomposition of parameters, we solve the constrained optimization subproblems set by set independently.

Specifically, let Q denote a parameter set. Since parameters from one baseline set share the same node i and the same parent configuration j, we use $\langle i, j\rangle$ to denote the index of a baseline set. A baseline set can be denoted as $Q = \{\langle i, j\rangle\}$, while a combined set, which consists of several baseline sets, can be denoted as $Q = \{\langle i, j\rangle, \langle i', j'\rangle, ...\}$.

The parameter learning problem can be decomposed into subproblems, one for each set of parameters. A subproblem can be formulated as follows:

$$\begin{aligned}
\text{Max} \quad & l_D(\theta) = \log \prod_{\langle i,j\rangle \in Q} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \\
\text{S.T.} \quad & g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0 \ \text{ for } \langle i, j\rangle \in Q \\
& h_p(\theta) \le 0 \ \ for\ 1 \le p \le S_Q
\end{aligned} \tag{9}$$

where g_{ij} represents an equality constraint, h_p represents an inequality constraint, S_Q is the number of inequality constraints in set Q.

Since the log likelihood function is concave, and the qualitative constraints are linear, Karush-Kuhn-Tucker (KKT) conditions [10] become sufficient to determine the solution to Eq.(9). The KKT conditions for the problem described in Eq.(9) are:

$$\nabla_\theta [l_D(\theta) - \sum_{\langle i,j\rangle \in Q} \lambda_{ij} g_{ij}(\theta) - \sum_{p=1}^{S_Q} \mu_p h_p(\theta)] = 0, \tag{10}$$

$$\begin{aligned}
g_{ij}(\theta) = 0, &\quad \text{for} \quad \langle i, j\rangle \in Q \\
h_p(\theta) \le 0, &\quad \text{for} \quad 1 \le p \le S_Q \\
\mu_p \ge 0, &\quad \text{for} \quad 1 \le p \le S_Q \\
\mu_p * h_p(\theta) = 0, &\quad \text{for} \quad 1 \le p \le S_Q
\end{aligned} \tag{11}$$

In optimization, an inequality constraint $h_p \le 0$ is active if $h_p = 0$, or inactive if $h_p < 0$. Based on this definition, we will derive closed form solutions for each type of constraints.

4.1 Range Constraints

Since range constraints (Eq.(4)) are applied to every individual parameters, we can solve the subproblems with range constraints within baseline sets. There are two constraints for each parameter θ_{ijk} in a baseline set $Q = \{\langle i, j\rangle\}$: $h_k^\alpha(\theta) = \theta_{ijk} - \alpha_{ijk} \le 0$ (upper bound constraint), and $h_k^\beta(\theta) = \beta_{ijk} - \theta_{ijk} \le 0$ (lower bound constraint).

As the objective function is concave and the range constraints are linear, the maximum solution either lies inside the feasible region defined by all constraints, when no constraint is active, or on the boundary defined by the active constrains, when some of the constraints are active. Assuming K_Q^β and K_Q^α are the sets of active constraints for lower bound and upper bound of parameters in Q respectively, and $K_Q = K_Q^\beta \cup K_Q^\alpha$ represents the set for all active constraints of parameters in Q, then the closed form solution for θ_{ijk} is as follows:

$$\theta_{ijk} = \begin{cases} \beta_{ijk} & \text{if } k \in K_Q^\beta \\ \alpha_{ijk} & \text{if } k \in K_Q^\alpha \\ (1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}) \dfrac{n_{ijk}}{\sum_{k \notin K_Q} n_{ijk}} & \text{otherwise} \end{cases} \tag{12}$$

Table 1: Search algorithm for finding active range constraints

Step 1:	Check the consistency of the range constraints: $0 < \alpha_{ijk} \leq 1$, $0 \leq \beta_{ijk} < 1$, $\alpha_{ijk} > \beta_{ijk}$, $\sum_{k=1}^{r_i} \beta_{ijk} \leq 1$, and $\sum_{k=1}^{r_i} \alpha_{ijk} \geq 1$ for $1 \leq k \leq r_i$. If satisfied, continue; else change constraints.
Step 2:	If $\sum_{k=1}^{r_i} \alpha_{ijk} = 1$, all the upper bound constraints should be active; else if $\sum_{k=1}^{r_i} \beta_{ijk} = 1$, all the lower bound constraints should be active. Else, continue.
Step 3:	Perform the ML estimation of parameters without constraints. Check the constraints with the estimated parameters $\theta^*_{ijk} = \frac{n_{ijk}}{N_{ij}}$. If no constraint is violated, then there is no active range constraint. Else, continue.
Step 4:	List all possible combinations of active constraints, and remove the combination if it contains more than $r_i - 1$ active constraints or $\sum_{k \in K_Q^\beta} \beta_{ijk} + \sum_{k \in K_Q^\alpha} \alpha_{ijk} \geq 1$.
Step 5:	For each of the remaining combination, compute λ_{ij}, until finding a λ_{ij} satisfying the criteria in Eq.(13).

The derivation is as follows. From the first equation of KKT conditions (Eq.(11)), we obtain $\theta_{ijk} = \frac{n_{ijk}}{\lambda_{ij} - \mu_k^\alpha + \mu_k^\beta}$. Because θ_{ijk} cannot be greater than α_{ijk} and less than β_{ijk} at the same time, at most one of the upper bound constraint h_k^α and lower bound constraint h_k^β for a parameter θ_{ijk} can be active at a time. Based on whether there is an active constraint for θ_{ijk}, two cases are considered.

- Case 1: If one of the upper bound and lower bound constraints is active, then $\theta_{ijk} = \alpha_{ijk}$, when $h_k^\alpha(\theta) = 0$; and $\theta_{ijk} = \beta_{ijk}$, when $h_k^\beta(\theta) = 0$.
- Case 2: If range constraints are not active, then $h_k^\alpha(\theta) < 0$, $h_k^\beta(\theta) < 0$ and $\mu_k^\alpha = \mu_k^\beta = 0$. Hence $\theta_{ijk} = \frac{n_{ijk}}{\lambda_{ij}}$. Summing up over all parameters whose constraints are not active, we get: $(1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}) = \sum_{k \notin K_Q} \theta_{ijk} = \frac{\sum_{k \notin K_Q} n_{ijk}}{\lambda_{ij}}$. Thus, we can obtain $\lambda_{ij} = \frac{\sum_{k \notin K_Q} n_{ijk}}{1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}}$, and $\theta_{ijk} = (1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}) \frac{n_{ijk}}{\sum_{k \notin K_Q} n_{ijk}}$, as shown in Eq.(12).

In this way, we derive the closed form solution for range constraints. To obtain solution in Eq.(12), we need to identify active constraints. Table 1 summarizes the algorithm to find active range constraints. The main idea of this algorithm is to search for the active constraints using the criteria in Eq.(13). Due to the page limit, we do not provide the proof for this equation.

$$\begin{cases} \lambda_{ij} \leq \frac{n_{ijk}}{\alpha_{ijk}} & k \in K_Q^\alpha \\ \lambda_{ij} \geq \frac{n_{ijk}}{\beta_{ijk}} & k \in K_Q^\beta \\ \lambda_{ij} \geq \frac{n_{ijk}}{\alpha_{ijk}},\ \lambda_{ij} \leq \frac{n_{ijk}}{\beta_{ijk}} & otherwise \end{cases} \tag{13}$$

4.2 Intra-Relationship Constraints

An Intra-relationship constraint defines the relationship between two parameters within one baseline set. Assuming parameters within one baseline set $Q = \{\langle i, j \rangle\}$ are $\theta_{ij1}, ...\theta_{ijr_i}$, which can be partitioned into $Q = A \cup B \cup C$, where $A = \{a_p | p = 1, 2, ..., S_Q\}$, $B = \{b_p | p = 1, 2, ..., S_Q\}$, such that $h_p(\theta) = \theta_{ija_p} - \theta_{ijb_p} \leq 0$, for $1 \leq p \leq S_Q$, and C is the set of parameters without intra-relationship constraints, the closed form solution for parameter $\theta_{ijk}$is as follows:

$$\theta_{ijk} = \begin{cases} \frac{n_{ija_p}+n_{ijb_p}}{2N_{ij}} & if\ k = a_p \text{ or } b_p \text{ and } n_{ija_p} \geq n_{ijb_p} \\ \frac{n_{ijk}}{N_{ij}} & Otherwise \end{cases} \tag{14}$$

where $N_{ij} = \sum_{k=1}^{r_i} n_{ijk}$. The derivation is similar to the one in Niculescu et al. [11].

4.3 Inter-Relationship Constraints

An Inter-relationship constraint defines the constraint applied on two parameters $\theta_{i'j'a}$ and $\theta_{i''j''b}$ from different baseline sets Q_A and Q_B, thus the subproblem for parameters with an inter-relationship constraint is applied on a combined parameter set $Q = Q_A \cup Q_B$, where baseline set $Q_A = \{\langle i', j' \rangle\}$ and baseline set $Q_B = \{\langle i'', j'' \rangle\}$, such that $h(\theta) = \theta_{i'j'a} - \theta_{i''j''b} \leq 0$. Let $N_A = \sum_{\langle i,j \rangle \in Q_A} n_{ijk}$, $N_B = \sum_{\langle i,j \rangle \in Q_B} n_{ijk}$, $n_a = n_{i'j'a}$, and $n_b = n_{i'j'b}$. The closed form solution for parameters with inter-relationship constraint is as follows. If $n_a N_B - N_A n_b \geq 0$

$$\theta_{ijk} = \begin{cases} \frac{n_a+n_b}{N_A+N_B} & ijk = i'j'a \text{ or } i''j''b \\ (1 - \frac{n_a+n_b}{N_A+N_B})\frac{n_{ijk}}{N_A-n_a} & \langle i,j \rangle \in Q_A\ and\ k \neq a \\ (1 - \frac{n_a+n_b}{N_A+N_B})\frac{n_{ijk}}{N_B-n_b} & \langle i,j \rangle \in Q_B\ and\ k \neq b \end{cases} \tag{15}$$

Else

$$\theta_{ijk} = \begin{cases} \frac{n_{ijk}}{N_A} & \langle i,j \rangle \in Q_A \\ \frac{n_{ijk}}{N_B} & \langle i,j \rangle \in Q_B \end{cases} \tag{16}$$

The brief derivation of the solution is as follows. The KKT conditions are:

$$\nabla_\theta[l_D(\theta) - \lambda_A g_A(\theta) - \lambda_B g_B(\theta) - \mu h(\theta)] = 0 \tag{17}$$

$$\begin{array}{ccc} g_A(\theta) = 0 & g_B(\theta) = 0 & \\ h(\theta) \leq 0 & \mu \geq 0 & \mu * h(\theta) = 0 \end{array} \tag{18}$$

From the first equation of KKT conditions (Eq.(18)), we can obtain:

$$\theta_{ijk} = \begin{cases} \frac{n_{ijk}}{\lambda_A+\mu} & ijk = i'j'a \\ \frac{n_{ijk}}{\lambda_B-\mu} & ijk = i''j''b \\ \frac{n_{ijk}}{\lambda_A} & \langle i,j \rangle \in Q_A\ and\ k \neq a \\ \frac{n_{ijk}}{\lambda_B} & \langle i,j \rangle \in Q_B\ and\ k \neq b \end{cases} \tag{19}$$

Two cases are considered, depending on whether the inter-relationship constraint is active or not:

- Case 1:$h(\theta) = 0$ and $\mu \geq 0$
  We can solve λ_A, λ_B, and μ with the following equations:

$$\begin{cases} \frac{n_a}{\lambda_A+\mu} = \frac{n_b}{\lambda_B-\mu} = \frac{n_a+n_b}{\lambda_A+\lambda_B} \\ \frac{n_a}{\lambda_A+\mu} + \frac{N_A-n_a}{\lambda_A} = 1 \\ \frac{n_b}{\lambda_B-\mu} + \frac{N_B-n_b}{\lambda_B} = 1 \end{cases} \tag{20}$$

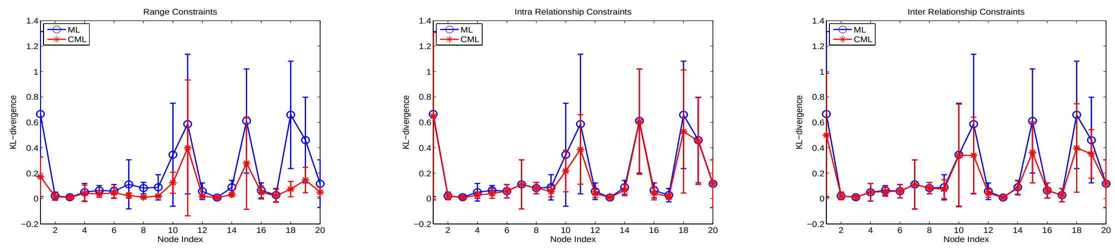
  The first equation is $h(\theta) = 0$, the second and the third are from $g_A(\theta) = 0$, and $g_B(\theta) = 0$. Also, from $\mu \geq 0$, we can get $n_a N_B - N_A n_b \geq 0$. In this way, we obtain the first part of closed form solution (Eq.(16)).

- Case 2: $h(\theta) < 0$ and $\mu = 0$
  It is equivalent to the case that no inequality constraints are applied. From $g_A(\theta) = 0$, we can get $\lambda_A = \sum_{\langle i,j \rangle \in Q_A} n_{ijk} = N_A$. Similarly, we can get $\lambda_B = N_B$. Plug them into Eq.(18), we can obtain the second part of closed form solution(Eq.(16)). From $h(\theta) < 0$, we get $n_a N_B - N_A n_b < 0$.

5 Evaluation with Synthetic Data

In order to test the performance of our method against ML estimation and the standard EM algorithm given sparse data and incomplete data respectively, we test the algorithms on multiple BNs with the same number of nodes of 20, but different randomly generated initial parameters and structures. For one specific BN structure, 11 BNs with different initializations of parameters are generated. One of them is treated as the ground truth, and 10 others as different initializations for parameter learning. For the case of sparse data, 700 samples are generated from the ground truth BN, 200 for testing data and the remaining 500 for training. For the case of incomplete data, 400 samples are drawn from the ground truth BN, half for training, half for testing, and all training data associated with hidden nodes are removed. To produce the needed constraints, for the case of sparse data, we randomly choose a subset of parameters from all parameters, and impose constraints on the selected parameters. For the case of incomplete data, we randomly choose parameters from only parameters for the hidden nodes, and impose constraints on them. The number of constraints in a CPT is no more than 2. For performance characterization, the Kullback-Leibler (K-L) divergence is used, which measures the distance between the learned parameters and the ground truth.

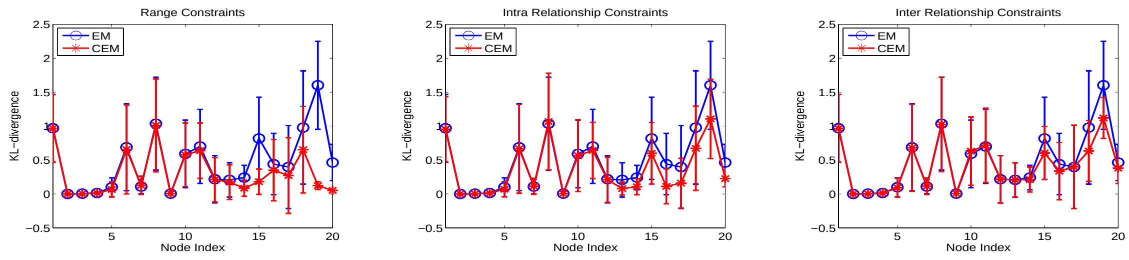
With complete but sparse data, we compare the learning performance of ML estimation with our method with range constraints, intra-relationship constraints and inter-relationship constraints respectively, as shown in Figure 1. We can see that CML is better than ML estimation in both mean and standard deviation of KL-divergence. More specifically, the mean K-L divergence for ML estimation is 0.2087, which decreases to 0.0786 for CML with range constraints, 0.1763 for CML with intra-relationship constraints, and 0.1546 for CML with inter-relationship constraints.



(a.1) (b.1) (c.1)

Figure 1: Sparse Data Learning Results Comparisons w.r.t K-L divergence: ML estimation vs. CML. (a) range constraints; (b) intra-relationship constraints; (c) inter-relationship constraints.

With incomplete data, we compare the learning performance of our method with standard EM method as shown in Figure 2. The average K-L divergence of hidden nodes decreases from 0.6437 for EM to 0.2361 for CEM with range constraints, 0.3830 for CEM with intra-relationship constraints, and 0.4864 for CEM with inter-relationship constraints. The improvements are especially significant for the hidden nodes (nodes 13 to 20).



(a.1) (b.1) (c.1)

Figure 2: Incomplete Data Learning Results Comparisons using w.r.t K-L divergence: EM vs. CEM. (a) range constraints; (b) intra-relationship constraints; (c) inter-relationship constraints.

6 Conclusion

Qualitative domain knowledge generally exists in applications. We define two types of constraints to represent the qualitative domain knowledge, and derive closed form solution for the maximum likelihood parameter estimator with the two types of constraints respectively. For the case of sparse data, we directly apply our constrained maximum likelihood estimator, while for incomplete data, we extend EM method by replacing M step with our constrained maximum likelihood estimator. The experimental results from synthetic data demonstrate that our method can fully exploit the domain knowledge to improve parameter learning accuracy.

References

[1] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *UAI*, pages 18–26, 2005.

[2] Cassio Polpo de Campos and Fabio Gagliardi Cozman. Belief updating and learning in semi-qualitative probabilistic networks. *UAI*, 2005.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *The Royal Statistical Society Series B*, 39:1–38, 1977.

[4] G. Elidan and N. Friedman. The information bottleneck em algorithm. *UAI*, pages 200–209, 2003.

[5] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. *AAAI*, pages 132–139, 2002.

[6] Ad Feelders and Linda van der Gaag. Learning bayesian network parameters under order constraints. *International Journal of Approximate Reasoning*, pages 37–53, 2006.

[7] David Heckerman. A tutorial on learning with bayesian networks. *M. Jordan, editor, learning in Graphic Models*. MIT Press, Cambridge, MA, 1999.

[8] M. Jaeger. The ai&m procedure for learning from incomplete data. *UAI*, pages 225–232, 2006.

[9] J.Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, pages 213–244, 1997.

[10] H. W. Kuhn and A. W. Tucker. Nonlinear programming. *Proc. of the second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492.

[11] Radu Stefan Niculescu, Tom M. Mitchell, and R. Bharat Rao. A theoretical framework for learning bayesian networks with parameter inequality constraints. *IJCAI*, 2007.

[12] F. Wittig and A. Jameson. Exploiting qualitative knowledge in the learning of conditional probabilities of bayesian networks. *UAI*, pages 644–652, 2000.

Modeling and Evaluating EBO-based Miltiary Planning using the Influence Diagram

Qiang Ji
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
jiq@rpi.edu

ARO Technical Report 2007-01
Intelligent Systems Laboratory
Dept. of Electrical, Computer, and Systems Eng.
Rensselaer Polytechnic Institute
August, 2007

Abstract

This report summarizes the preliminary work we have done to evaluate the effectivness of modeling and evaluating an EBO-based military planning using the Influence Diagram (ID). It includes construction of an ID model, its parameterization, and evaluation of the model using different action-selection strategies.

1 Introduction

A military plan includes various military actions, the effects of these actions, as well as sensory observations that assess the effectiveness of the actions. In addition, a miliary plan also includes its goal as well as subgoals that are needed to achieve the goal. For effects-based military planning, we need a model that can model different factors, their uncertainties, and their dependencies. In addition, the model should have a mechanism that can propagate the action effects, their uncertainties, and their dynamics. To meet these requirements, a probabilistic framework based on the dynamic Influence Diagram (DID) is used.

An Influence Diagram(ID) is a directed acyclic graph which consists of random nodes, decision nodes, and the value (utility) nodes. Here, the sensors and the actions are represented by the decision nodes. The action effects, the goal and subgoals, and the observations are represented by the random nodes. The utility of a sensor or an action is represented by the utility nodes. Figure 1 shows a generic ID for EBO-based planing modeling. Figure 2 shows the application of the model to a specific military planning problem.

2 Model Description

2.1 Structure of the Model

Figure 3 shows an example ID structure for a generic military planning problem. The planning problem consists of 4 military actions, a goal, and several subgoals. In total, there are 33 nodes in this ID, and all the random nodes and decision nodes are binary. Specifically, circular nodes represent the random variables, rectangular nodes are the decision variables, and the diamond nodes are the utility variables. G node depicts the goal of the military plan

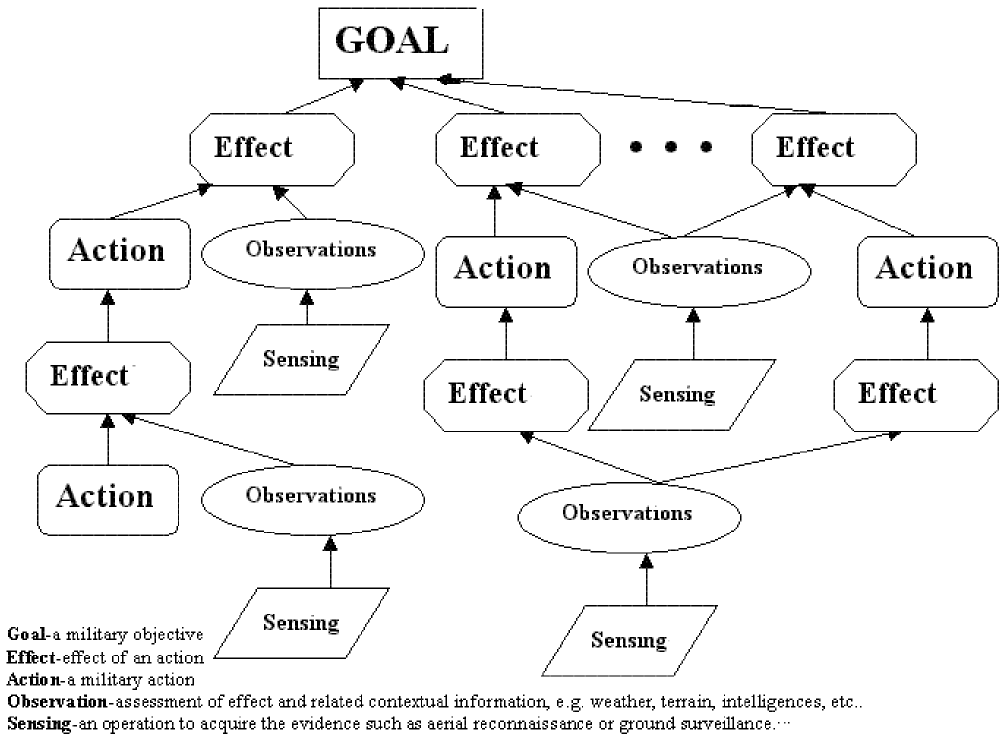

GOAL
Effect
Effect
Effect
Action
Observations
Action
Observations
Action
Effect
Sensing
Effect
Sensing
Effect
Action
Observations
Observations
Sensing
Sensing
Goal-a military objective
Effect-effect of an action
Action-a military action
Observation-assessment of effect and related contextual information, e.g. weather, terrain, intelligences, etc..
Sensing-an operation to acquire the evidence such as aerial reconnaissance or ground surveillance.

Figure 1: A generic influence diagram for EBO modeling

to achieve. G_0 node is the goal node in the last time slice. SG and SSG represent the sub-goal and sub-subgoal that are needed to achieve the goal. The action effects are represented by the E nodes. The A nodes represents the military actions, the sensing operations that assess the miliary action effects are represented by the S nodes. The sensory observations resulted from the sensing operations are represented by the O nodes. Finally, the diamond U nodes represent the action utilities.

2.2 Parameters of the Model

Given the topology of the model, the next task it to parameterize the model. Model parametrization requires to specify the conditional probabilities of each node given each configuration of its parents. Model parametrization

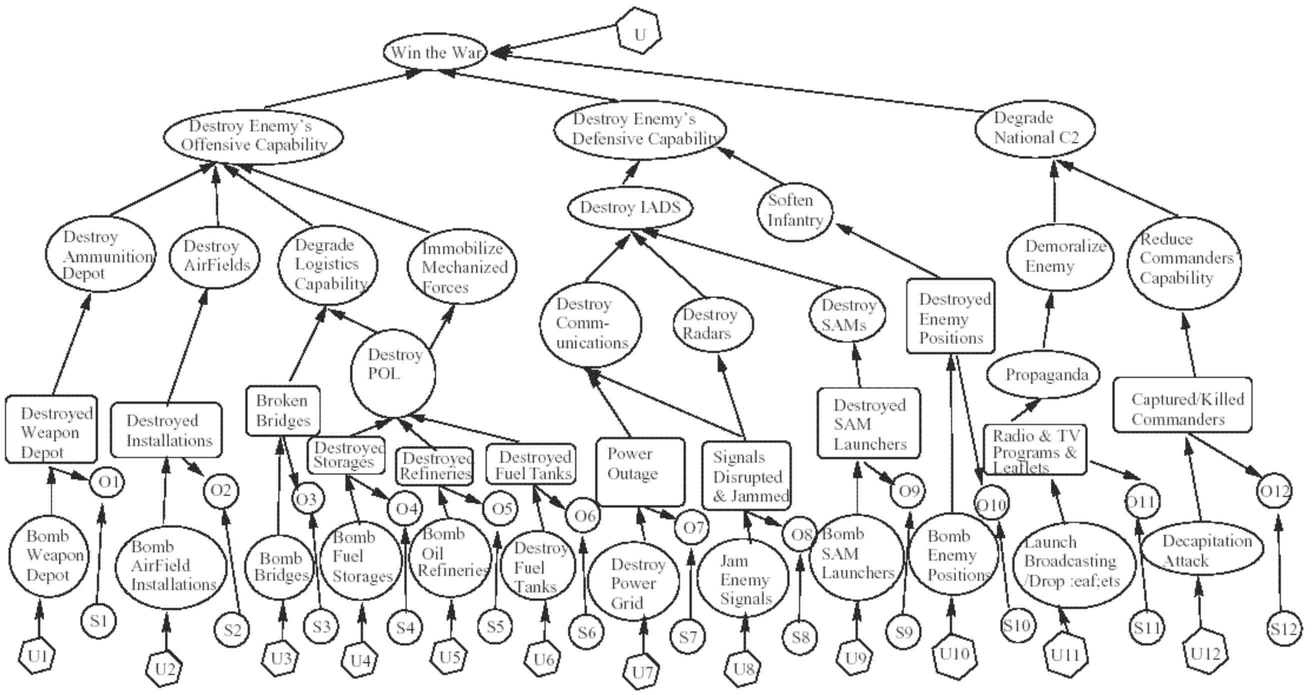


Figure 2: An example DID for EBO-based military plan modeling and assessment, where the circular nodes at the bottom represent various actions and the rectangular nodes the action effects.

can be done automatically through a learning method or manually by a domain expert. Here, we manually set the parameters.

Specifically, Table 1 summarizes the conditional probability tables (CPTs) of the model shown in Fig. 3. Table 1 (a) provides the CPTs for the effect nodes E, table 1 (b) provides the CPTs for the observation nodes O, and table 1 (c) consists of the CPTs for the goal node. Each node has two values, with 1 being false and 2 being true. In addition, a random number r between -0.05 and 0.05 is added to the probabilities to vary the effects of different actions.

We also need specify the parameters for the utility nodes. Table 2 shows the utility functions associated with the action and sensor nodes. The utility

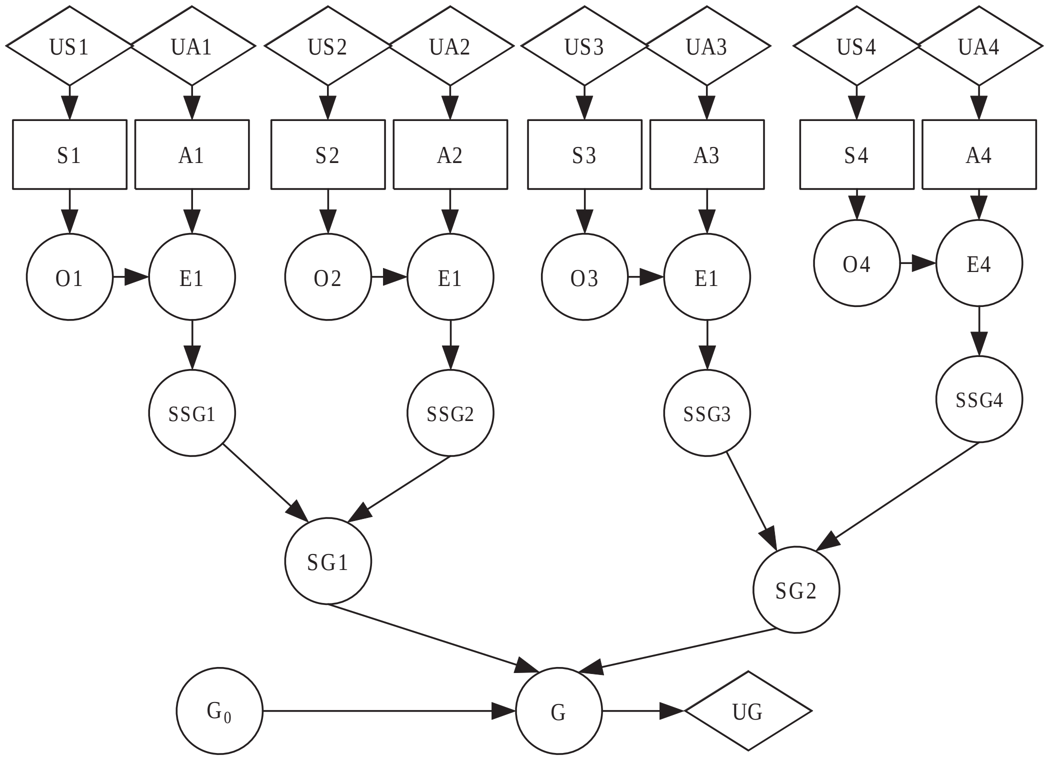


Figure 3: A Dynamic Influence Diagram model for a generic military planning.

of a military action is measured by its contribution to achieving the military goal or subgoal while the utility of a sensing action is measured by the tradeoff between its cost and its contribution to understanding the effect of an action. These utility values vary from action to action and they also vary over time. For this study, their values are randomly assigned.

The parameters of all the nodes are assigned based on the following heuristic rules:

1. If there is no action, then there is no effect. Thus, when an A node equals to false, the probability of the corresponding E node being True is zero.

2. We assume that the sensors can effectively detect the effect of an action,

which means that if there is an effect, the probability of observation being true is 0.7.

3. With the accomplishment of both subgoals, the chance of goal node equals to true is also high. On the other hand, if either of the subgoals is achieved, the chance of goal node being true is low. As the two subgoals may be of different importance in accomplishing the goal, the probability of goal given each subgoal is, therefore, different.

4. After every time slice, the probability of the goal node at next time slice will be initialized as the value of G_0 node. At the initial time slice, the probability of G_0 node being true is 0.5.

3 Model Evaluation

Figure 4 provides the flowchart that we use to evaluate the model. Basically, the process repeats like this. At first, we initialize the prior probability of goal attainment to be 0.5, and then begin to iteratively choose the best plan until the probability of achieving the goal is high enough or the net expected utility of the selected plan is too low. The two thresholds can be set according to the real conditions.

During each iteration, if the probability of achieving the goal is below a threshold, we proceed to identify a military plan that can maximally improve the chance of the goal attainment. Once the plan is identified, the actions of the plan are executed, and the action effects are propagated. This is then followed by activating the corresponding sensory operations to assess the effects the actions. The acquired sensory observations are propagated through the model to update its joint probabilities. This propagation also updates the probability of the goal node. The process then repeats. In summary, the process includes the following steps:

- The first step is to infer probability of goal node to decide whether to take further actions or to stop.

- If in the first step, we did not stop, we identify the best plan, execute its constituent actions, and propagate their effects.

Table 1: Conditional Probability Tables: (a) effect nodes; (b) observation nodes; (c) the goal node

$P(E_i\|A_i)(i = 1, 2, 3, 4)$	$E_i = 1$	$E_i = 2$
$A_i = 1$	1	0
$A_i = 2$	$1 - P(E_i = 2\|A_i = 2)$	0.7+r

(a)

$P(O\|E)$	O=1	O=2
E=1	0.8	0.2
E=2	0.3	0.7

(b)

	$P(G = 2\|SG1, SG2, G_0)$
(2,2,2)	0.99
(1,2,1)	0.65
(2,1,2)	0.97
(2,1,1)	0.58
(1,1,2)	0.89
(2,2,1)	0.81
(1,2,2)	0.92
(1,1,1)	0.01

(c)

Table 2: Utility functions: (a)and (b) action utilties with respect to goal; (c) costs of sensing operations.

	U
G=1	0
G=2	10

(a)

	U(A_i)
$A_i = 1(i = 1, 2, 3, 4)$	0
$A_i = 2(i = 1, 2, 3, 4)$	rand(0,1)

(b)

	U(S_i)
$S_i = 1(i = 1, 2, 3, 4)$	0
$S_i = 2(i = 1, 2, 3, 4)$	rand(0,1)

(c)

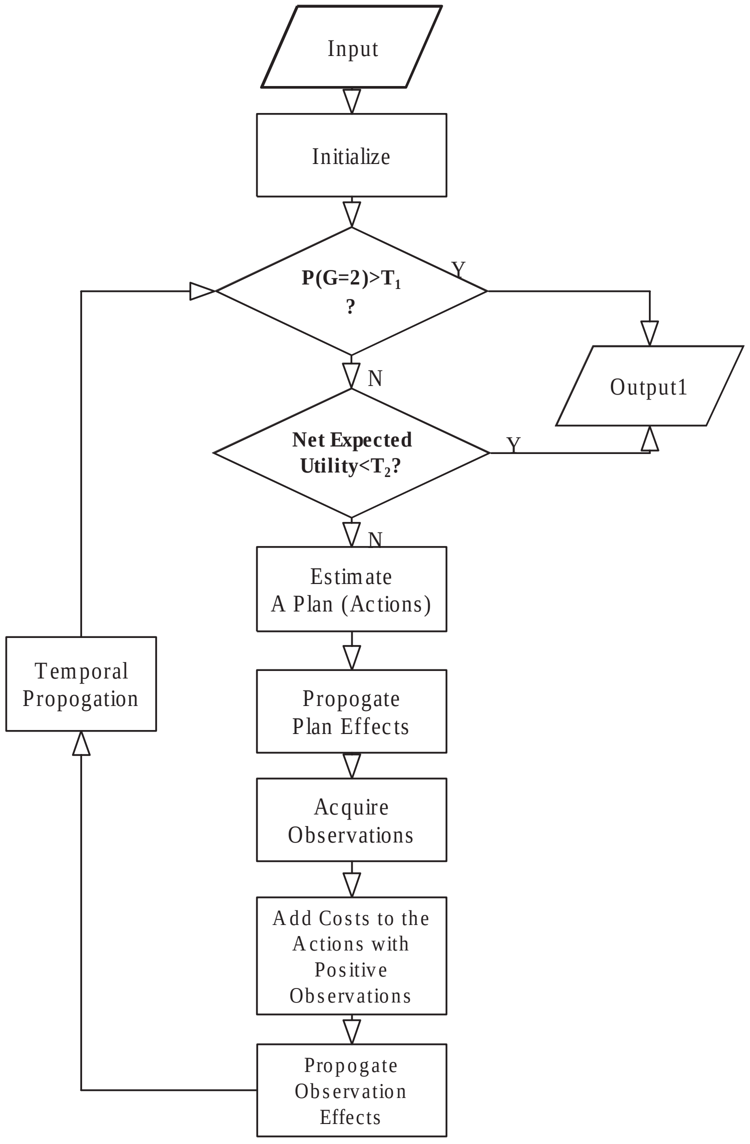


Figure 4: Flowchart of the algorithm

- The third step is to obtain the observations, which provides an assessment of the results of action effects, and to propagate the observations to update the goal attainment probability.

4 Action Selection Methods

To form a military plan, we need identify its constituent actions. There are three kinds of criteria, on which actions are selected.

1. Only one action is chosen each time. Two algorithms are implemented here. Algorithm brute force 1 chooses the best action which maximizes the net expected utility(i.e.,$E_\delta(U)$ in Eq. 1). Algorithm random 1 randomly chooses one action each time.

2. Obtain a plan consisting of a set of actions, that maximizes the net expected utility without considering action costs. Three algorithms are implemented here: brute force, greedy, and random selection.

3. Obtain a plan to maximize the net expected utility while the cost of the plan is under a given budget limit. Three algorithms are implemented here: brute force, simple selection, and the random selection. All are subject to an upper budget limit.

The expected utility of a plan is defined as

$$E_\delta = E_\delta(U) - \Sigma_{i=1}^{n} U(A_i) \tag{1}$$

where the first item represents the net expected utility (Eq. 2) of the plan δ, which consists of a series of actions. The second item is the cost of the plan, which consists of costs of all the actions involved in the plan. n is the number of actions that can be chosen in a plan.

$$E_\delta(U) = \sum_{i=1}^{2} P(G = i|\delta) * U(G = i) \tag{2}$$

4.1 One Action for Each Plan

4.1.1 Brute Force 1

Brute force 1 finds the best action based on the net expected utility. Table 3 summarizes the algorithm.

Table 3: Brute Force Action Selection Method

Step 1:	List all the candidate actions.
Step 2:	Compute the net expected utility for each action.
Step 3:	Find the best action which maximizes $E_\delta(U)$.

4.1.2 Random 1

Random 1 randomly selects an action from the candidate actions every time.

4.2 Multiple Actions for Each Plan: No Cost

4.2.1 Brute Force

Brute force can find the optimal plan with the largest net expected utility. There is no limit on the number of actions in a plan. The algorithm is described in Table 4.

Table 4: Brute Force Action Selection Method

Step 1:	List all the possible plans (combinations of actions).
Step 2:	Compute the net expected utility for each combination.
Step 3:	Find the best plan which maximizes $E_\delta(U)$

4.2.2 Multiple Actions for Each Plan: Greedy Method

For a large model with many action nodes, it will take much time to find the global optimal plan using a brute force method. Greedy method, which costs less time, is an alternative to brute force method to find a local optimal plan efficiently. Table 5 summarizes the greedy action selection algorithm.

Table 5: Greedy Action Selection Method

Step 1:	Compute the net expected utility for each solo action.
Step 2:	Find the best action which maximizes the net expected utility, and add it to a plan.
Step 3:	Find an action in the remaining actions which can maximize the net expected utility, when combined with the chosen actions as a new plan
Step 4:	This process repeats until the net expected utility of the plan peaks.

4.2.3 Multiple Actions for Each Plan: Random Selection

To compare with the brute force and the greedy method, we also implement the random selection method. For the random selection method, we confine the maximum number of actions selected by the random algorithm to the maximum number of actions selected by the brute force algorithm. Table 6 summarizes the process of random selection.

Table 6: Random Action Selection Method

Step 1:	Identify the number of actions selected by the bruce force method.
Step 2:	Randomly choose the same number of actions to form a plan.

4.3 Multiple Actions for Each Plan with a Budget limit

4.3.1 Brute Force with a budget limit

Brute force with a budge limit, which is described in Table 7, finds a plan to maximize the net expected utility subject to a budget limit.

4.3.2 Simple Method with a budget limit

Simple method with a budget limit simply chooses the first several actions whose total cost is under the given budget limit. The algorithm is summa-

Table 7: Brute Force Action Selection Method

Step 1:	List all the possible plans (combinations of actions).
Step 2:	Compute the cost of all the plans respectively. If the cost of a plan is beyond the given budget limit, remove it from the possible plans.
Step 2:	Compute the net expected utility for the remaining plans respectively.
Step 3:	Find the best plan which maximizes the net expected utility.

rized in Table 8.

Table 8: Simple Action Selection Method with a Budget Limit

Step 1:	Compute the net expected utility for each solo action, and sort the actions in descending order.
Step 2:	Choose the first several actions whose sum of costs is under the given budget limit.

4.3.3 Random Selection with a budget limit

Random selection with a budget limit randomly selects a plan whose cost is under the budget limit. The algorithm is summarized in Table 9.

Table 9: Random Action Selection Method

Step 1:	Randomly choose several actions as a plan.
Step 2:	Check whether the cost of the plan is beyond the budget limit. If so, go to step 1. Otherwise terminate the program and output the plan.

5 Experiment Results

To test the various action selection methods mentioned above, we conducted three experiments. All the experiments are based on the model shown in Fig. 3. Since the observation nodes are instantiated randomly, the probability of goal attainment also varies. Hence, we perform each experiment 20 times to obtain the average.

5.1 Experiment 1: one action for each plan

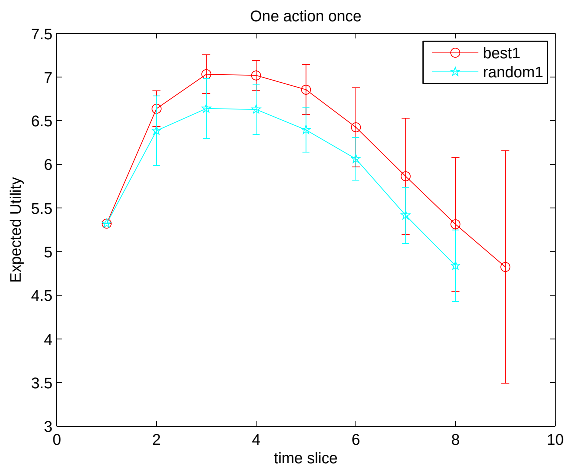
In the first experiment, one action is chosen every time. The experiment results are shown in Fig. 5. Figure 5 (a) is the net expected utility of the selected plan in each time slice, and (b) is the corresponding probability of goal attainment. The median of each bar is the mean value, and the height of the bar is the standard deviation, which is obtained from 20 experimental results.

The algorithm marked by 'best1' is brute force 1 algorithm, which chooses one best action based on the net expected utility criterion, and the algorithm marked by 'random1' chooses one action randomly. We can see from the figures that the mean of the net expected utility of brute force 1 is greater than random 1 in each time slice except for the start point, when they are equal.

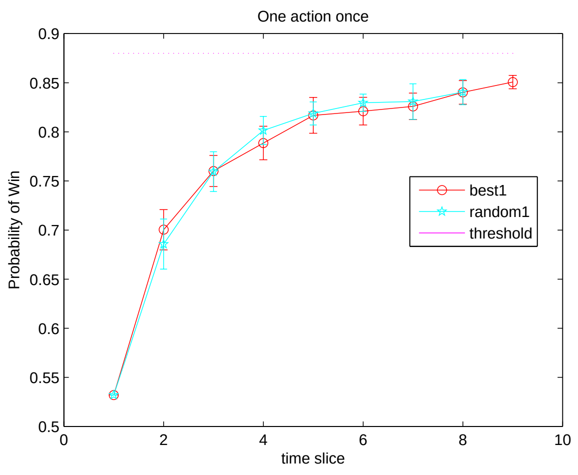
5.2 Experiment 2: Multiple Actions without Cost

In this experiment, we test the algorithm brute force, greedy and random action selection. The first two algorithms choose actions based on the net expected utility. The random selection selects actions randomly. We confine the maximum number of actions selected by the random algorithm to be the same as the maximum number of actions selected by the brute force algorithm.

Figure 6 shows the experiment results. Figure 6 (a) is the net expected utility in each time slice, and (b) is the corresponding probability of goal attainment. We can see from the figures that the mean net expected utility of the plan using the brute force and greedy methods is greater than that of the random method in each time slice except for the starting point, when they are all the same. The experiment result of brute force is close to that



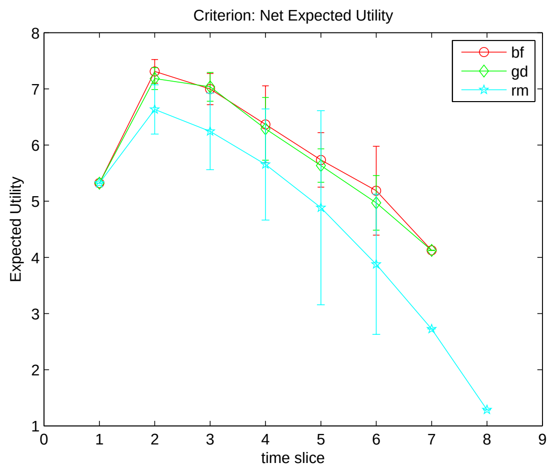
(a)



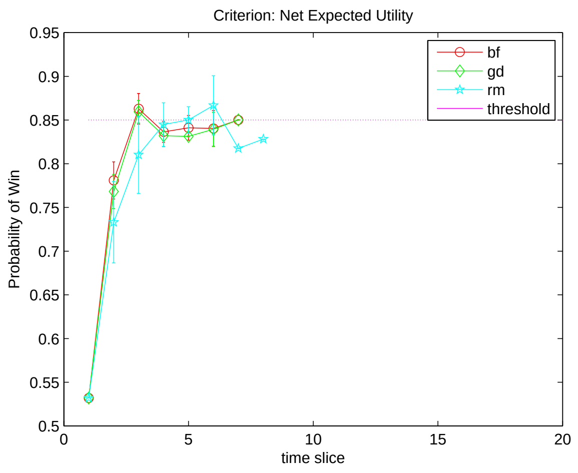
(b)

Figure 5: Experiment 1 Results: (a) Expected utility of the selected plan at each time instant; (b) Probability of goal attainment after executing the selected plan at each time instant.

of greedy algorithm, while in some time slice, it is a little better than the greedy method. Computationally however, the greedy method is much faster than the brute force method. Finally, the height of the error bar, which represents the standard derivation, shows the impact of observation variation on the experimental results. If the observations are false, it may decrease the value of a plan, and on the other hand, if they are true, it may increase the performance of a plan.



(a)



(b)

Figure 6: Experiment 2 Results: (a) Expected utility of the selected plan, (b) Probability of goal attainment after executing the selected plan at each time instant.

5.3 Experiment 3: Multiple Actions with Cost

In this experiment, we test the brute force, simple method and random action selection methods, all under a budget limit on cost of the selected plan. The fist two algorithms choose actions to maximize the net expected utility given a budget limit. The random selection randomly selects a plan under the given budget limit. Figure 7 shows the experiment results. Similar to the experiment results above, Figure 7 (a) is the net expected utility of the selected plan for each time slice, and (b) is the corresponding probability of goal attainment.

To maximize the expected utility, we are to maximize the probability of goal attainment. We can see from (b) that both simple method and brute force perform are much better than the random method in goal attainment. Brute force with a budget limit achieves the threshold most quickly, followed by the simple method. Random method with a budget limit takes the longest time to reach the threshold, and sometimes it never achieves it. The simple method, compared with the brute force method, is computationally much more efficient.

Table 10 shows the steps taken by the three algorithms respectively in one experiment. Index is the number of time slice. Index 0 is the initial point. Actions are the index of actions chosen in the time slice. Observations are the observation results. 0 means no observation, 1 means false and 2 means true.

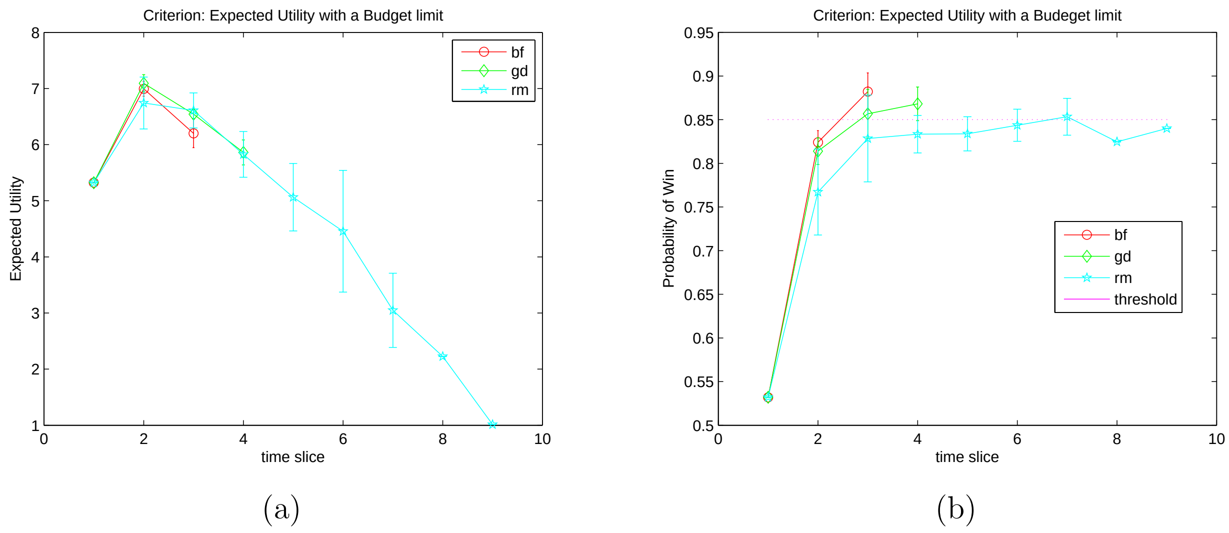


Figure 7: Experiment 3 Results: (a) Expected utility of the selected plan; (b) Probability of goal attainment

Table 10: Budge limit cases: (a) Brute force method; (b) Simple method; (c) Random selection method

Index	Actions	Observations	$P(G=2)$
0	[]	[0 0 0 0]	0.532
1	[1 3 4]	[1 0 1 2]	0.805
2	[1 2 3]	[2 2 1 0]	0.900

(a)

Index	Actions	Observations	$P(G=2)$
0	[]	[0 0 0 0]	0.532
1	[1 4 2]	[1 2 0 2]	0.8284
2	[1 3]	[1 0 1 0]	0.8476
3	[1 3]	[1 0 2 0]	0.8826

(b)

Index	Actions	Observations	$P(G=2)$
0	[]	[0 0 0 0]	0.532
1	[2 3]	[0 1 2 0]	0.7505
2	[2 4]	[0 2 0 1]	0.8456
3	[4]	[0 0 0 2]	0.849
4	[4]	[0 0 0 2]	0.8495
5	[2]	[0 1 0 0]	0.8315
6	[1]	[2 0 0 0]	0.8531

(c)

5.4 Conclusion

This preliminary study demonstrates the promise of the influence diagram for EBO-based miliary modeling and assessment. Specifically, for modeling, an ID can effectively represent actions, their effects, their relationships, and their uncertainties. In addition, through belief propagation, action effects and their uncertainties can be systematically propagated through the model.

For plan assessment, we perform three experiments to evaluate the performance of three different action selection strategies in terms of their optimality and efficiency in action selection. Specifically, in the first two experiments, net expected utility is used as the criterion, so the experiment results show that the action selection methods, such as brute force and greedy method, are better than the random algorithm in net expected utility in each time slice. But their performance with respect to the goal attainment appear to be comparable. This needs to be further investigated. In the third experiment, the net expected utility is used with a budget limit. As the utility function for the goal never changes over time, the probability of goal attainment is maximized with the maximization of the expected utility. From the experiment 3 results, we can conclude that that both brute force with a budget limit and simple method perform better than random selection with a budget limit, helping achieving the plan goal faster.

While the experiments identify a few issues, the experimental results remain preliminary, and further studies and analysis will be needed.

6 Appendix: An efficient method for computing the expected utility

In this section, we try to find a method to improve computational complexity of the action selection algorithm. Regardless of the action selection method we use (except for the random method), we need the probability of goal attainment to compute the expected utility or the net expected utility. In the action selection methods described above, we need to perform inference in every time slice to obtain the probability of goal attainment. But if we can find a more efficient way to compute the probability of goal attainment, we may decrease the computational complexity of the action selection method.

A general ID for military models is shown in Figure 8. There are n actions and m subgoals in the model. The structure of model inside the dotted lines can be arbitrary, which means we will not consider the specific number of nodes or lines in the region.

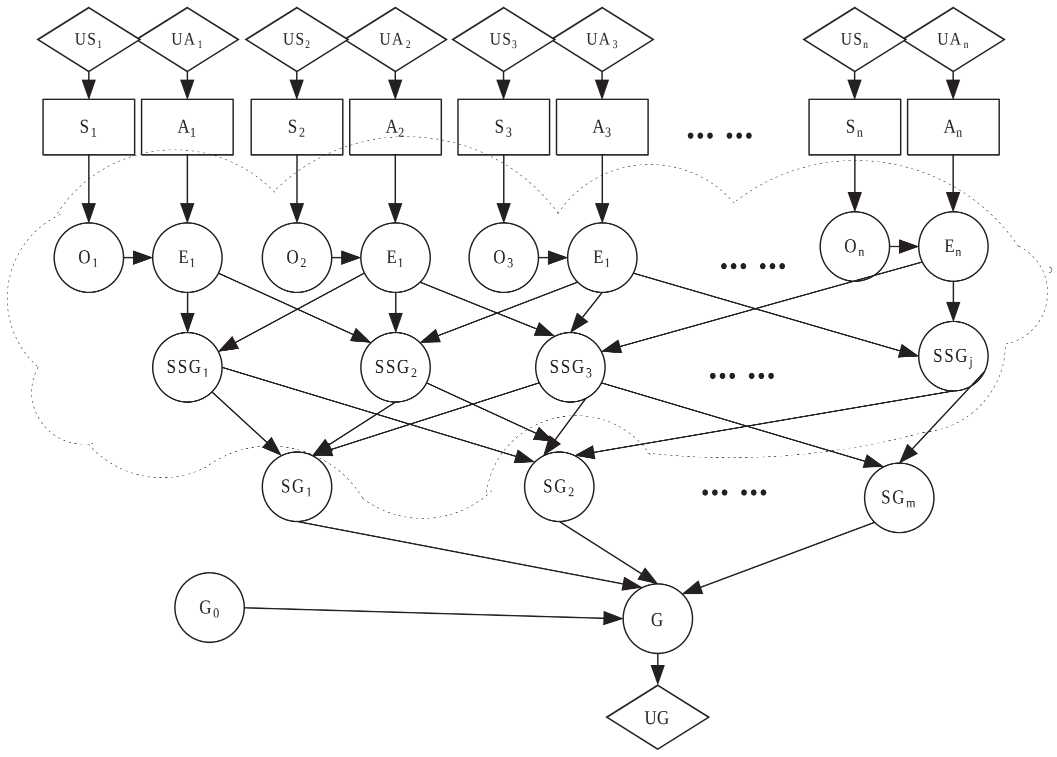


Figure 8: A general ID for military plan modeling

As we can see from the general military model above, the only parameter that changes over time is the probability of goal in last time slice. As the structure and other parameters remain the same over time, we may factorize the constant coefficients for the fixed structure so that they only need be computed once.

Specifically, given the structure described in Figure 8, we can derive the probability of goal attainment as follows:

$$\begin{aligned}
P(G=2|A) &= \sum_{G_0=1}^{2} P(G|G_0,A)P(G_0|A) \\
&= \sum_{G_0=1}^{2} P(G|G_0,A)P(G_0) \\
&= \sum_{G_0=1}^{2} \sum_{SG} P(G|G_0,SG)P(G_0)P(SG|A) \\
&= x * \sum_{SG} P(G|G_0=2,SG) * P(SG|A) \\
&+ (1-x) * \sum_{SG} P(G|G_0=1,SG) * P(SG|A) \qquad (3)
\end{aligned}$$

where,

$$\begin{aligned}
A &= \{A_1, ..., A_n\} \\
SG &= \{SG_1, ..., SG_m\} \\
x &= P(G_0=2)
\end{aligned}$$

where A represents all the action variables. SG represents all the subgoals connected to the goal. n and m are the total number of actions and the number of subgoals respectively.

Based on the equations above, we can see that the probability of goal attainment given different actions only depends on x, because other factors in the equation are constant, which will not change over time. We can only compute each combination of the constant factor once when we first use them, and use them again later without recomputing them again. This will lead to computational saving.

A Factorization Approach for Efficient EBO-based Military Plan Assessment

Qiang Ji, Ph. D
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
jiq@rpi.edu

ARO Technical Report 2007-02
Intelligent Systems Laboratory
Dept. of Electrical, Computer, and Systems Eng.
Rensselaer Polytechnic Institute
May, 2007

1 Introduction

An important issue in effects-based operation (EBO) is evaluation of the effects of a military plan. A military plan consists of a set of selected actions and their execution timings. For each available action, a planner must decide whether to include it in the plan. In making these choices, the planner's objective is to identify a plan that maximizes the probability of achieving the military objectives while minimizing the associated costs. As the number of possible plans increases exponentially with the number of actions, determining the optimal plan via an exhaustive search of the plan space becomes computationally intractable and practically infeasible. We propose a factorization procedure to significantly reduce the evaluation time for each plan by exploiting the common computations associated with evaluating each plan. Experimental study of our method shows that it can often reduce the evaluation time by orders of magnitude. In addition, the proposed method offers exact solution, therefore avoiding the convergence problem plaguing the commonly employed approximate inference methods such as logic sampling.

2 Proposed Solution

We model the military planning problem using an Influence Diagram (ID) as shown in Figure 1. Different from causal model in [1], an influence diagram explicitly includes the action nodes and the value nodes. The primary military actions are represented by the rectangular nodes X_i ($i \in \{1, \ldots, n\}$) while the top node H represents the military goal. Each action node X_i is associated with a value node U_i, encoding the cost of performing the action. The cost for an action includes physical cost, collateral damages, political cost, etc.. The goal node H is also associated with a value node U, encoding the value of goal attainment. In general, the actions do not influence the overall goal directly; instead their consequences propagate through a set of intermediate subgoals/tasks whose realization lead to the goal success. In an ID, these subgoals/tasks are represented by the intermediate circular nodes $Y_j^i (j \in \{1, \ldots, m_i\})$, which represents the j-th subgoal at level i [1] The links in an ID specify three classes of probabilistic relations: the relation between the action nodes and lowest-level subgoals, the relation among the many subgoals, and the relation between the highest-level subgoals and the overall goal.

Given such an ID, for each action node X_i, a plan δ prescribes an action choice $\delta(X_i)$ of a_i or $\neg a_i$, where a_i stands for the selection of action X_i and $\neg a_i$ represents otherwise. Thus, a plan δ can be denoted by $\delta = \{\delta(X_1), \ldots, \delta(X_n)\}$. Let $g(U_i|\delta(X_i))$ and $g(U|H)$ be the cost of performing action X_i and the value of goal attainment respectively, then the expected net utility of executing a plan δ is

$$E_\delta = E_\delta[U] - \sum_{i=1}^{n} g(U_i|\delta(X_i)). \tag{1}$$

where the first term, $E_\delta[U] = \sum_H P(H|\delta)g(U|H)$, represents the expected utility of the goal and the second term is the combined cost of executing the selected actions. The optimal plan δ^* is the one that maximizes E_δ over all plans, i.e.,

[1] For simplicity, we assume that the subgoal nodes are structured in three levels, although our factorization approach generalizes to arbitrary levels.

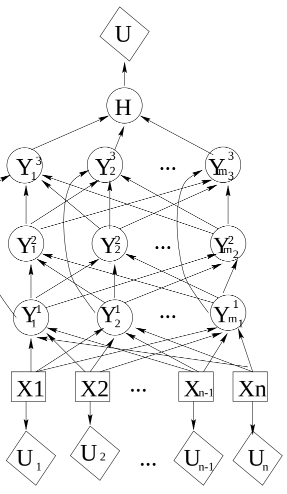


Figure 1: A casual influence diagram modeling the military planning problem, where squares are action nodes, circles represent subgoals/tasks and the goal, and diamonds are value nodes.

$$\delta^* = \arg\max_\delta E_\delta \tag{2}$$

The brute-force approach requires to evaluate all 2^n plans for n actions to identify the optimal plan. In this report, we present an approach that can significantly reduce the evaluation time for each plan.

3 A Factorization Approach to Efficient Plan Evaluation

We propose a factorization procedure that allows to efficiently evaluate each plan by exploiting the common computations in the network. The method allows a plan to be evaluated often in orders of magnitude time less than the normal evaluation. Specifically, with respect to the ID in Figure 1, $E_\delta[U]$ can be computed as

$$E_\delta[U] = \sum_H \sum_{Y^1_{1:m_1}} \sum_{Y^2_{1:m_2}} \sum_{Y^3_{1:m_3}} \Pi_{j_1=1}^{m_1} P(Y^1_{j_1}|\delta) \Pi_{j_2=1}^{m_2} P(Y^2_{j_2}|Y^1_{1:m_1}) \Pi_{j_3=1}^{m_3} P(Y^3_{j_3}|Y^2_{1:m_2}, Y^1_{j_3}) P(H|Y^3_{1:m_3}) g(U|H).$$

where $Y^i_{1:m_i}$ denotes $\{Y^i_1, \ldots, Y^i_{m_i}\}$ for the nodes at the i-th level.

Exchanging the summing order among variables, we have

$$E_\delta[U] = \sum_{Y^1_{1:m_1}} \begin{array}{l} \left[\sum_H \sum_{Y^2_{1:m_2}} \sum_{Y^3_{1:m_3}} \Pi_{j_2=1}^{m_2} P(Y^2_{j_2}|Y^1_{1:m_1}) \Pi_{j_3=1}^{m_3} P(Y^3_{j_3}|Y^2_{1:m_2}, Y^1_{j_3}) P(H|Y^3_{1:m_3}) g(U|H)\right] \\ \left[\Pi_{j_1=1}^{m_1} P(Y^1_{j_1}|\delta)\right] \end{array} \tag{3}$$

Let us define two functions f_U and f_δ corresponding to the terms within the two brackets.

$$\begin{array}{l} f_U = \sum_H \sum_{Y^2_{1:m_2}} \sum_{Y^3_{1:m_3}} \Pi_{j_2=1}^{m_2} P(Y^2_{j_2}|Y^1_{1:m_1}) \Pi_{j_3=1}^{m_3} P(Y^3_{j_3}|Y^2_{1:m_2}, Y^1_{j_3}) P(H|Y^3_{1:m_3}) g(U|H) \\ f_\delta = \Pi_{j_1=1}^{m_1} P(Y^1_{j_1}|\delta) \end{array} \tag{4}$$

With these two functions, we rewrite $E_\delta[U]$ in Equation (3) as

$$E_\delta[U] = \sum_{Y_1^1,\ldots,Y_{m_1}^1} f_U f_\delta. \tag{5}$$

It can be seen that the function f_U is independent of the plan δ while the function f_δ is dependent on the plan δ. In other words, the computations for f_U need to be computed only once but can be used across all plans. These computations can be *factored out.* The factorization plan evaluation can be implemented as follows

```
1. Pre-compute the quantities $f_U$ for all plans
2. For each plan $\delta$
   compute $f_\delta$ for each assignment of $Y^1_{1:m_1}$
   compute $E_\delta[U]$ by Equation (5)
3. Return the plan $\delta^*$ that maximizes $E_\delta[U]$
```

Table 1: The factorization approach to ID evaluation

4 Experiments

We have conducted experiments to evaluate the performance of the factorization approach. In our experiments, we use the IDs in Figure 1 as the test example. The CPTs are randomly generated. The value functions for value nodes are manually specified. The code is written in Matlab V6.5 and runs in a laptop with a 2.0 GHz CPU under Windows XP. In our experiments, we compare the factorization approach against the generic brute-forced approach. For convenience, we refer to them respectively as `evalCS` (named after Computation Sharing) and `evalBF` (named after Brute-Forced).

To see how the performance of the algorithms vary with the number of action nodes, we fix the number of subgoal nodes at each level at four and vary the number of action nodes. Thus the static ID with n action nodes has additionally 10 random nodes and n+1 value nodes. We ran `evalBF` and `evalCS` for seven problems with $n = 3, 5, ..., 13$. The timing data are presented in left chart of Figure 2. The chart gives the total CPU seconds that the algorithms took for each of the problems. Note that the vertical axis is drawn in log-scale. The solid (dashed) curve is for `evalCS(evalBF)`. It can be seen that `evalCS` is considerably more efficient than `evaBF`. For instance, by our collected data, for $n = 9$, to evaluate 512 plans, `evalCS` took 3.51 seconds while `evalBF` 646.74 seconds; for $n = 13$, to evaluate 8192 plans, `evalCS` used 46.32 seconds while `evalBF` 9284.05 seconds.

5 Conclusions and Future Work

In this report, we proposed a factorization approach to significantly reduce the time for the evaluation of a military plan. Our approach identifies the common computations used by all

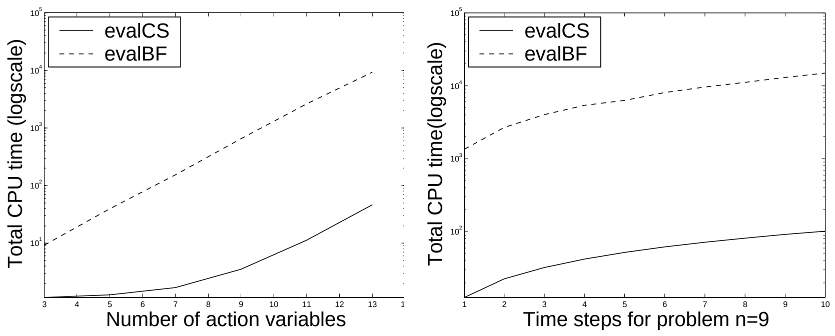


Figure 2: Performance comparison of evalBF and evalCS

plans and factors these computations out so that they only need be computed once. This has resulted in a significant computational saving, often by several orders of magnitude. Experimental results validate our method.

Despite these successes, there remains work to do for the factorization procedure. So far, our method is only applied to static ID. Further investigations, theoretical developments, and experimental validations are necessary to extend our method to handle dynamic network, network with evidences, and other complex network structures including those highly non-layered networks and those with utilities attached to the intermediate nodes. In addition, we need develop the theories for analytically determining the upper and lower bound performance for a given network, without explicitly evaluating any plan. This allows the planner to commence plan evaluation to identify the best one only if the upper bound and lower bound performance exceeds his/her expectation.

References

[1] U. Kuter, D. Nau, and J. F. Lemmer. Interactive planning under uncertainty with causal modeling and abalysis. Technical Report CS-TR-4434, Department of Computer Science, University of Marryland, 2003.

Exploiting Qualitative Constraints for Learning Bayesian Networks under Insufficient Data

Zheng Xue and Qiang Ji
Rensselaer Polytechnic Institute

Abstract

Graphical models (GMs) such as Bayesian Networks (BN) or the Influence Diagrams (ID) are being increasingly applied to many different applications. One bottleneck in using GMs is that learning the GM model parameters often requires a relative large amount of training data. However, in real life and for many applications, training data is often incomplete or sparse, which can cause low learning accuracy. Incorporating domain knowledge can help alleviate this problem. Instead of using quantitative prior knowledge as used by most of the existing methods, this paper introduces a novel learning method based on systematically combining the training data with some qualitative knowledge.

To validate our method, we compare it with the Maximum Likelihood (ML) estimation method under sparse data and with the Expectation Maximization (EM) algorithm under incomplete data respectively. The experimental results show that our method improves the parameter learning accuracy significantly compared with both ML and EM algorithms.

1 Introduction

Among all the issues of graphical models, parameter learning is one of the main challenges. Parameter learning is to estimate the entries of the conditional probability distributions (CPDs) given the structure of a model. Many learning techniques rely heavily on training data [7]. Ideally, with sufficient data, it is possible to learn the parameters by standard statistical analysis like maximum likelihood (ML) estimation. In many real-world cases, however, the data are either incomplete or sparse, which can cause inaccurate parameter estimation. Data incompleteness is defined as missing of data for some parameters, while data sparseness means the amount of training data is limited.

When data are incomplete, Expectation-Maximization (EM) [3] algorithm is often used. Most EM-based methods work under the assumption that data are missing at random (MAR), which means the missing values can be estimated by the observed ones in some way. However, when data are missing completely at random (MCAR), e.g data of hidden nodes, the learned parameters could be far from the ground truth. The reason is that the missing data do not even depend on the observed ones, and there is no way to estimate the missing data only from the observed ones.

In our paper, we propose a framework to solve the parameter learning problem by combining quantitative data and domain knowledge in the form of qualitative constraints. Two kinds of qualitative constraints are defined: range constraints which are applied to individual parameters; and relationship constraints which are applied to pairs of parameters. For sparse but complete data, we solve the learning task by reformulating the problem as a constrained ML (CML) problem. For incomplete data, we introduce the constrained EM (CEM) by adding constraints to the M step, and iteratively solve the learning problem. In addition, we provide closed form solutions to both CML and CEM.

2 Related Work

We have already discussed that one of the shortcomings of EM algorithms is that it can easily be trapped in a local maximum when data are MCAR. Till now, there are many different methods to help EM to escape from the local maximum, such as the information-bottleneck EM algorithm [4], data perturbing method [5], and AI&M procedure [8]. These methods focus on improving the machine learning techniques, but ignoring the useful domain knowledge.

Domain knowledge can be classified as quantitative and qualitative knowledge, which describe the explicit quantification of parameters, and approximate characterizations of parameters respectively. Both kinds of domain knowledge are useful for parameter learning. While the quantitative knowledge has been widely used in the form of prior probability distributions, qualitative constraints have not been fully exploited in parameter learning yet.

Wittig et al. [12] present a method to integrate qualitative constraints into two learning algorithms, APN [9] and EM, by adding violation functions as a penalty term to the log likelihood function. They show that domain knowledge in the form of constraints can improve learning accuracy. However, this penalty-based method cannot guarantee to find the global maximum. Besides, the weights for the penalty functions often need be manually tuned, depending on applications. Altendorf et al. [1] describes a method to incorporate monotonicity constraints into learning algorithm. It is based on the assumption that the values of the variables can be totally ordered. Additionally, it also uses the penalty functions, which suffers from the same problem as [12]. Feelders and Van der Gaag [6] incorporate some simple inequality constraints in the learning process. They assume that all the variables are binary. The constraints used in the above methods [1, 12, 6] are restrictive, as each constraint has to involve all parameters in a conditional probability table (CPT).

Campos and Cozman [2] formulate the learning problem as a constrained optimization problem. However, they do not provide a specific method to solve the optimization problem. Niculescu et al. [11] also solve the learning problem by optimization techniques. They derive the closed form solutions with ML estimation for two kinds of constraints: inequalities between sums of parameters and upper bounds on sum of parameters within a CPT. There are two main limitations of their method: First, they assume one parameter can and only can have one constraint, and there is no overlap between parameters of different constraints. Second, their method cannot handle constraints from different CPTs. We improve their method by deriving the closed form solution for range constraints, which contain both upper bound and lower bound constraints for the same parameters. In addition, the relationship constraints defined in our paper can either be within or between CPTs.

3 Problem Definition and Approach

3.1 Basic Parameter Learning Theory

We focus on parameter learning in a Bayesian Networks with all discrete nodes, where the structure is known in advance. The method can be extended to other graphical models including the IDs. The notations are defined as follows. Assume a BN with n nodes, θ is the entire vector of parameters, and θ_{ijk} denotes one of the parameters. $\theta_{ijk} = p(x_i^k|pa_i^j)$, where i $(i = 1, ..., n)$ ranges over all the variables in the BN, j $(j = 1, ..., q_i)$ ranges over all the possible parent configurations of node (variable) X_i, and k $(k = 1, ..., r_i)$ ranges over all the possible states of X_i. Therefore, x_i^k represents the kth state of node X_i, and pa_i^j is the jth parent configuration of node X_i.

Given a dataset $D = \{D_1, ..., D_N\}$, which consists of samples of the BN nodes, the goal of parameter learning is to find the most probable values $\hat{\theta}$ for θ that can best explain the dataset D, which is usually quantified by the log likelihood function $\log(p(D|\theta))$, denoted as $L_D(\theta)$. Assuming that the examples are drawn independently from the underlying distribution, based on the conditional independence assumptions in BNs, we have the log likelihood function in Eq.(1), where n_{ijk} is the count for the case that node i has the state k, with the state configuration j for its parent nodes.

$$L_D(\theta) = \log \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{n_{ijk}} \tag{1}$$

If the dataset D is complete, ML estimation method can be described as a constrained optimization problem, i.e. maximize (Eq.(2)), subject to n equality constrains (Eq.(3)).

$$\begin{array}{ll} \text{Max} & L_D(\theta) \qquad (2) \\ \text{S.T.} & g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0 \qquad (3) \end{array}$$

where g_{ij} imposes the constraint that each parameter sums to 1 over all its state, $1 \leq i \leq n$ and $1 \leq j \leq q_i$.

If dataset D is incomplete, ML estimation cannot be applied directly. A common method is standard EM algorithm [3], which starts from some initial point, and then iteratively takes E step and M step to get a local maximum of the likelihood function. Particularly for discrete nodes, E step computes the expected counts for all parameters, and M step estimates the parameters by maximizing log likelihood function, given the counts from E step. EM algorithm can guarantee to converge to a local maximum. However, depending on different initializations, it may converge to different local maxima. When there are a large number of missing data, which means there are many local maxima, EM algorithm can get stuck in a local maximum far away from the global one.

3.2 Qualitative Constraints

We introduce two kinds of qualitative constraints, which can be easily specified by domain experts. They are range and relationship constraints.

Range constraint defines the upper bound and lower bound of some parameters. Assuming α_{ijk} and β_{ijk} are the upper bound and lower bound for parameter θ_{ijk}, then the range constraints can be defined as follows:

$$\beta_{ijk} \leq \theta_{ijk} \leq \alpha_{ijk} \qquad (4)$$
$$\text{where} \quad 0 < \alpha_{ijk} \leq 1 \text{ and } 0 \leq \beta_{ijk} < 1$$

Relationship constraint defines the relative relationship between a pair of parameters. If both of the two parameters in a relationship constraint share the same node index i, and parent configuration j, the constraint is called *intra-relationship constraint*, which can be represented as follows:

$$\theta_{ijk} \leq \theta_{ijk'} \text{ where } k \neq k' \qquad (5)$$

If the two parameters in a relative relationship constraint do not satisfy the requirement of an intra-relationship constraint, the constraint is called *inter-relationship constraint*. It can be described as follows:

$$\theta_{ijk} \leq \theta_{i'j'k'} \text{ where } i \neq i' \text{ or } j \neq j' \qquad (6)$$

3.3 Overview of Our Approach

We aim to solve the learning problem by reformulating the problem as a constrained based optimization problem, i.e.,

$$\begin{array}{lll} \text{Max} & L_D(\theta) & \qquad (7) \\ \text{S.T.} & g_{ij}(\theta) = \sum_{k=1}^{r_i} \theta_{ijk} - 1 = 0, & 1 \leq i \leq n, \text{ and } 1 \leq j \leq q_i \\ & h_p(\theta) \leq 0, & 1 \leq p \leq S \end{array}$$

where $h_p(x) \leq 0$ denotes the inequality constraints, and S is the total number of inequality constraints. Using the Lagrange multipliers λ_{ij} and μ_p, the objective function to be maximized can be incorporated with the constraints, producing the following augmented objective function

$$f(\theta) = L_D(\theta) - \sum_{i=1}^{n} \sum_{j=1}^{q_i} \lambda_{ij} g_{ij}(\theta) - \sum_{k=1}^{S} \mu_p h_p(\theta) \qquad (8)$$

Given Eq.(8), for sparse but complete data, we can directly apply the CML method by maximizing Eq.(8) to estimate the parameters. For incomplete data, we can replace the M step of EM algorithm by the solution to Eq.(8), and iteratively obtain the estimation of the parameters. In the section to follow, we introduce our solution to Eq.(8).

4 Parameter Learning With Qualitative Constraints

In this section, we derive the closed form solutions for maximizing Eq.(8) under different types of constraints. Because of the decomposability of the log likelihood function, we can deal with small independent optimization subproblems on independent parameter sets separately instead of dealing with all parameters simultaneously. For this, we define two kinds of parameter sets: one is the *baseline set*, which contains parameters with the same node and the same parent configuration; the other is the *combined set*, which contains several baseline sets. We first separate parameters into baseline sets, and then if there is a constraint on parameters from different baseline sets, we combine those baseline sets into one new combined set. This process continues until there is no constraint on parameters from different sets. After decomposition of parameters, we solve the constrained optimization subproblems set by set independently.

Specifically, let Q denote a parameter set. Since parameters from one baseline set share the same node i and the same parent configuration j, we use $\langle i,j\rangle$ to denote the index of a baseline set. A baseline set can be denoted as $Q=\{\langle i,j\rangle\}$, while a combined set, which consists of several baseline sets, can be denoted as $Q=\{\langle i,j\rangle,\langle i',j'\rangle,...\}$.

The parameter learning problem can be decomposed into subproblems, one for each set of parameters. A subproblem can be formulated as follows:

$$\begin{array}{ll} \text{Max} & l_D(\theta)=\log\prod_{\langle i,j\rangle\in Q}\prod_{k=1}^{r_i}\theta_{ijk}^{n_{ijk}} \\ \text{S.T.} & g_{ij}(\theta)=\sum_{k=1}^{r_i}\theta_{ijk}-1=0 \;\text{ for }\; \langle i,j\rangle\in Q \\ & h_p(\theta)\le 0 \;\; for\; 1\le p\le S_Q \end{array} \tag{9}$$

where g_{ij} represents an equality constraint, h_p represents an inequality constraint, S_Q is the number of inequality constraints in set Q.

Since the log likelihood function is concave, and the qualitative constraints are linear, Karush-Kuhn-Tucker (KKT) conditions [10] become sufficient to determine the solution to Eq.(9). The KKT conditions for the problem described in Eq.(9) are:

$$\nabla_\theta[l_D(\theta)-\sum_{\langle i,j\rangle\in Q}\lambda_{ij}g_{ij}(\theta)-\sum_{p=1}^{S_Q}\mu_p h_p(\theta)]=0, \tag{10}$$

$$\begin{array}{rll} g_{ij}(\theta)=0, & \text{for} & \langle i,j\rangle\in Q \\ h_p(\theta)\le 0, & \text{for} & 1\le p\le S_Q \\ \mu_p\ge 0, & \text{for} & 1\le p\le S_Q \\ \mu_p * h_p(\theta)=0, & \text{for} & 1\le p\le S_Q \end{array} \tag{11}$$

In optimization, an inequality constraint $h_p\le 0$ is active if $h_p=0$, or inactive if $h_p<0$. Based on this definition, we will derive closed form solutions for each type of constraints.

4.1 Range Constraints

Since range constraints (Eq.(4)) are applied to every individual parameters, we can solve the subproblems with range constraints within baseline sets. There are two constraints for each parameter θ_{ijk} in a baseline set $Q=\{\langle i,j\rangle\}$: $h_k^\alpha(\theta)=\theta_{ijk}-\alpha_{ijk}\le 0$ (upper bound constraint), and $h_k^\beta(\theta)=\beta_{ijk}-\theta_{ijk}\le 0$ (lower bound constraint).

As the objective function is concave and the range constraints are linear, the maximum solution either lies inside the feasible region defined by all constraints, when no constraint is active, or on the boundary defined by the active constrains, when some of the constraints are active. Assuming K_Q^β and K_Q^α are the sets of active constraints for lower bound and upper bound of parameters in Q respectively, and $K_Q=K_Q^\beta\cup K_Q^\alpha$ represents the set for all active constraints of parameters in Q, then the closed form solution for θ_{ijk} is as follows:

$$\theta_{ijk}=\begin{cases} \beta_{ijk} & \text{if } k\in K_Q^\beta \\ \alpha_{ijk} & \text{if } k\in K_Q^\alpha \\ (1-\sum_{k\in K_Q^\beta}\beta_{ijk}-\sum_{k\in K_Q^\alpha}\alpha_{ijk})\dfrac{n_{ijk}}{\sum_{k\notin K_Q}n_{ijk}} & \text{otherwise} \end{cases} \tag{12}$$

Table 1: Search algorithm for finding active range constraints

Step 1:	Check the consistency of the range constraints: $0 < \alpha_{ijk} \leq 1$, $0 \leq \beta_{ijk} < 1$, $\alpha_{ijk} > \beta_{ijk}$, $\sum_{k=1}^{r_i} \beta_{ijk} \leq 1$, and $\sum_{k=1}^{r_i} \alpha_{ijk} \geq 1$ for $1 \leq k \leq r_i$. If satisfied, continue; else change constraints.
Step 2:	If $\sum_{k=1}^{r_i} \alpha_{ijk} = 1$, all the upper bound constraints should be active; else if $\sum_{k=1}^{r_i} \beta_{ijk} = 1$, all the lower bound constraints should be active. Else, continue.
Step 3:	Perform the ML estimation of parameters without constraints. Check the constraints with the estimated parameters $\theta^*_{ijk} = \frac{n_{ijk}}{N_{ij}}$. If no constraint is violated, then there is no active range constraint. Else, continue.
Step 4:	List all possible combinations of active constraints, and remove the combination if it contains more than $r_i - 1$ active constraints or $\sum_{k \in K_Q^\beta} \beta_{ijk} + \sum_{k \in K_Q^\alpha} \alpha_{ijk} \geq 1$.
Step 5:	For each of the remaining combination, compute λ_{ij}, until finding a λ_{ij} satisfying the criteria in Eq.(13).

The derivation is as follows. From the first equation of KKT conditions (Eq.(11)), we obtain $\theta_{ijk} = \frac{n_{ijk}}{\lambda_{ij} - \mu_k^\alpha + \mu_k^\beta}$. Because θ_{ijk} cannot be greater than α_{ijk} and less than β_{ijk} at the same time, at most one of the upper bound constraint h_k^α and lower bound constraint h_k^β for a parameter θ_{ijk} can be active at a time. Based on whether there is an active constraint for θ_{ijk}, two cases are considered.

- Case 1: If one of the upper bound and lower bound constraints is active, then $\theta_{ijk} = \alpha_{ijk}$, when $h_k^\alpha(\theta) = 0$; and $\theta_{ijk} = \beta_{ijk}$, when $h_k^\beta(\theta) = 0$.
- Case 2: If range constraints are not active, then $h_k^\alpha(\theta) < 0$, $h_k^\beta(\theta) < 0$ and $\mu_k^\alpha = \mu_k^\beta = 0$. Hence $\theta_{ijk} = \frac{n_{ijk}}{\lambda_{ij}}$. Summing up over all parameters whose constraints are not active, we get: $(1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}) = \sum_{k \notin K_Q} \theta_{ijk} = \frac{\sum_{k \notin K_Q} n_{ijk}}{\lambda_{ij}}$. Thus, we can obtain $\lambda_{ij} = \frac{\sum_{k \notin K_Q} n_{ijk}}{1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}}$, and $\theta_{ijk} = (1 - \sum_{k \in K_Q^\beta} \beta_{ijk} - \sum_{k \in K_Q^\alpha} \alpha_{ijk}) \frac{n_{ijk}}{\sum_{k \notin K_Q} n_{ijk}}$, as shown in Eq.(12).

In this way, we derive the closed form solution for range constraints. To obtain solution in Eq.(12), we need to identify active constraints. Table 1 summarizes the algorithm to find active range constraints. The main idea of this algorithm is to search for the active constraints using the criteria in Eq.(13). Due to the page limit, we do not provide the proof for this equation.

$$\begin{cases} \lambda_{ij} \leq \frac{n_{ijk}}{\alpha_{ijk}} & k \in K_Q^\alpha \\ \lambda_{ij} \geq \frac{n_{ijk}}{\beta_{ijk}} & k \in K_Q^\beta \\ \lambda_{ij} \geq \frac{n_{ijk}}{\alpha_{ijk}},\ \lambda_{ij} \leq \frac{n_{ijk}}{\beta_{ijk}} & otherwise \end{cases} \tag{13}$$

4.2 Intra-Relationship Constraints

An Intra-relationship constraint defines the relationship between two parameters within one baseline set. Assuming parameters within one baseline set $Q = \{\langle i, j \rangle\}$ are $\theta_{ij1}, ...\theta_{ijr_i}$, which can be partitioned into $Q = A \cup B \cup C$, where $A = \{a_p | p = 1, 2, ..., S_Q\}$, $B = \{b_p | p = 1, 2, ..., S_Q\}$, such that $h_p(\theta) = \theta_{ija_p} - \theta_{ijb_p} \leq 0$, for $1 \leq p \leq S_Q$, and C is the set of parameters without intra-relationship constraints, the closed form solution for parameter $\theta_{ijk}$is as follows:

$$\theta_{ijk} = \begin{cases} \frac{n_{ija_p}+n_{ijb_p}}{2N_{ij}} & if\ k = a_p \text{ or } b_p \text{ and } n_{ija_p} \geq n_{ijb_p} \\ \frac{n_{ijk}}{N_{ij}} & Otherwise \end{cases} \tag{14}$$

where $N_{ij} = \sum_{k=1}^{r_i} n_{ijk}$. The derivation is similar to the one in Niculescu et al. [11].

4.3 Inter-Relationship Constraints

An Inter-relationship constraint defines the constraint applied on two parameters $\theta_{i'j'a}$ and $\theta_{i''j''b}$ from different baseline sets Q_A and Q_B, thus the subproblem for parameters with an inter-relationship constraint is applied on a combined parameter set $Q = Q_A \cup Q_B$, where baseline set $Q_A = \{\langle i', j' \rangle\}$ and baseline set $Q_B = \{\langle i'', j'' \rangle\}$, such that $h(\theta) = \theta_{i'j'a} - \theta_{i''j''b} \leq 0$. Let $N_A = \sum_{\langle i,j \rangle \in Q_A} n_{ijk}$, $N_B = \sum_{\langle i,j \rangle \in Q_B} n_{ijk}$, $n_a = n_{i'j'a}$, and $n_b = n_{i'j'b}$. The closed form solution for parameters with inter-relationship constraint is as follows. If $n_a N_B - N_A n_b \geq 0$

$$\theta_{ijk} = \begin{cases} \frac{n_a+n_b}{N_A+N_B} & ijk = i'j'a \text{ or } i''j''b \\ (1 - \frac{n_a+n_b}{N_A+N_B})\frac{n_{ijk}}{N_A-n_a} & \langle i,j \rangle \in Q_A\ and\ k \neq a \\ (1 - \frac{n_a+n_b}{N_A+N_B})\frac{n_{ijk}}{N_B-n_b} & \langle i,j \rangle \in Q_B\ and\ k \neq b \end{cases} \tag{15}$$

Else

$$\theta_{ijk} = \begin{cases} \frac{n_{ijk}}{N_A} & \langle i,j \rangle \in Q_A \\ \frac{n_{ijk}}{N_B} & \langle i,j \rangle \in Q_B \end{cases} \tag{16}$$

The brief derivation of the solution is as follows. The KKT conditions are:

$$\nabla_\theta[l_D(\theta) - \lambda_A g_A(\theta) - \lambda_B g_B(\theta) - \mu h(\theta)] = 0 \tag{17}$$

$$\begin{matrix} g_A(\theta) = 0 & g_B(\theta) = 0 & \\ h(\theta) \leq 0 & \mu \geq 0 & \mu * h(\theta) = 0 \end{matrix} \tag{18}$$

From the first equation of KKT conditions (Eq.(18)), we can obtain:

$$\theta_{ijk} = \begin{cases} \frac{n_{ijk}}{\lambda_A+\mu} & ijk = i'j'a \\ \frac{n_{ijk}}{\lambda_B-\mu} & ijk = i''j''b \\ \frac{n_{ijk}}{\lambda_A} & \langle i,j \rangle \in Q_A\ and\ k \neq a \\ \frac{n_{ijk}}{\lambda_B} & \langle i,j \rangle \in Q_B\ and\ k \neq b \end{cases} \tag{19}$$

Two cases are considered, depending on whether the inter-relationship constraint is active or not:

- Case 1:$h(\theta) = 0$ and $\mu \geq 0$
  We can solve λ_A, λ_B, and μ with the following equations:

$$\begin{cases} \frac{n_a}{\lambda_A+\mu} = \frac{n_b}{\lambda_B-\mu} = \frac{n_a+n_b}{\lambda_A+\lambda_B} \\ \frac{n_a}{\lambda_A+\mu} + \frac{N_A-n_a}{\lambda_A} = 1 \\ \frac{n_b}{\lambda_B-\mu} + \frac{N_B-n_b}{\lambda_B} = 1 \end{cases} \tag{20}$$

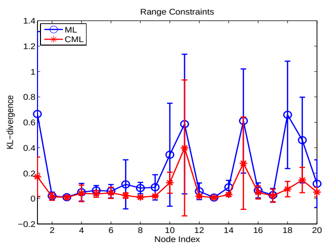
  The first equation is $h(\theta) = 0$, the second and the third are from $g_A(\theta) = 0$, and $g_B(\theta) = 0$. Also, from $\mu \geq 0$, we can get $n_a N_B - N_A n_b \geq 0$. In this way, we obtain the first part of closed form solution (Eq.(16)).

- Case 2: $h(\theta) < 0$ and $\mu = 0$
  It is equivalent to the case that no inequality constraints are applied. From $g_A(\theta) = 0$, we can get $\lambda_A = \sum_{\langle i,j \rangle \in Q_A} n_{ijk} = N_A$. Similarly, we can get $\lambda_B = N_B$. Plug them into Eq.(18), we can obtain the second part of closed form solution(Eq.(16)). From $h(\theta) < 0$, we get $n_a N_B - N_A n_b < 0$.

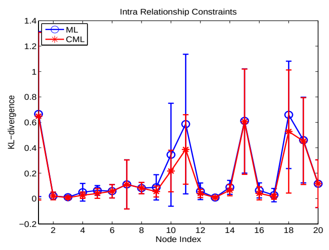
5 Evaluation with Synthetic Data

In order to test the performance of our method against ML estimation and the standard EM algorithm given sparse data and incomplete data respectively, we test the algorithms on multiple BNs with the same number of nodes of 20, but different randomly generated initial parameters and structures. For one specific BN structure, 11 BNs with different initializations of parameters are generated. One of them is treated as the ground truth, and 10 others as different initializations for parameter learning. For the case of sparse data, 700 samples are generated from the ground truth BN, 200 for testing data and the remaining 500 for training. For the case of incomplete data, 400 samples are drawn from the ground truth BN, half for training, half for testing, and all training data associated with hidden nodes are removed. To produce the needed constraints, for the case of sparse data, we randomly choose a subset of parameters from all parameters, and impose constraints on the selected parameters. For the case of incomplete data, we randomly choose parameters from only parameters for the hidden nodes, and impose constraints on them. The number of constraints in a CPT is no more than 2. For performance characterization, the Kullback-Leibler (K-L) divergence is used, which measures the distance between the learned parameters and the ground truth.

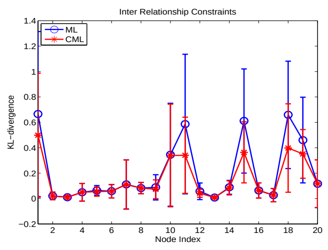
With complete but sparse data, we compare the learning performance of ML estimation with our method with range constraints, intra-relationship constraints and inter-relationship constraints respectively, as shown in Figure 1. We can see that CML is better than ML estimation in both mean and standard deviation of KL-divergence. More specifically, the mean K-L divergence for ML estimation is 0.2087, which decreases to 0.0786 for CML with range constraints, 0.1763 for CML with intra-relationship constraints, and 0.1546 for CML with inter-relationship constraints.



(a.1)



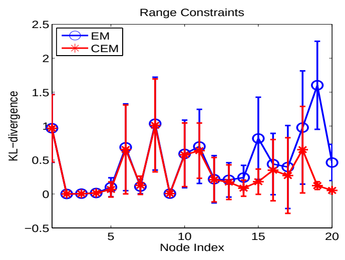
(b.1)



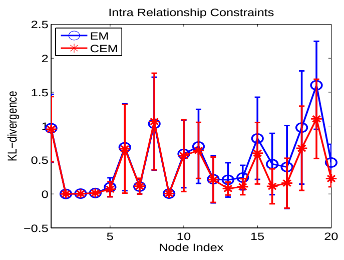
(c.1)

Figure 1: Sparse Data Learning Results Comparisons w.r.t K-L divergence: ML estimation vs. CML. (a) range constraints; (b) intra-relationship constraints; (c) inter-relationship constraints.

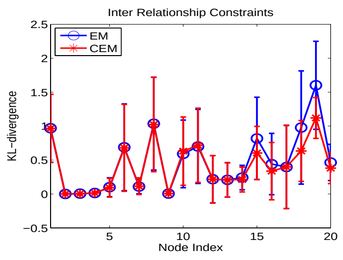
With incomplete data, we compare the learning performance of our method with standard EM method as shown in Figure 2. The average K-L divergence of hidden nodes decreases from 0.6437 for EM to 0.2361 for CEM with range constraints, 0.3830 for CEM with intra-relationship constraints, and 0.4864 for CEM with inter-relationship constraints. The improvements are especially significant for the hidden nodes (nodes 13 to 20).



(a.1)



(b.1)



(c.1)

Figure 2: Incomplete Data Learning Results Comparisons using w.r.t K-L divergence: EM vs. CEM. (a) range constraints; (b) intra-relationship constraints; (c) inter-relationship constraints.

6 Conclusion

Qualitative domain knowledge generally exists in applications. We define two types of constraints to represent the qualitative domain knowledge, and derive closed form solution for the maximum likelihood parameter estimator with the two types of constraints respectively. For the case of sparse data, we directly apply our constrained maximum likelihood estimator, while for incomplete data, we extend EM method by replacing M step with our constrained maximum likelihood estimator. The experimental results from synthetic data demonstrate that our method can fully exploit the domain knowledge to improve parameter learning accuracy.

References

[1] Eric E. Altendorf, Angelo C. Restificar, and Thomas G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *UAI*, pages 18–26, 2005.

[2] Cassio Polpo de Campos and Fabio Gagliardi Cozman. Belief updating and learning in semi-qualitative probabilistic networks. *UAI*, 2005.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *The Royal Statistical Society Series B*, 39:1–38, 1977.

[4] G. Elidan and N. Friedman. The information bottleneck em algorithm. *UAI*, pages 200–209, 2003.

[5] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. *AAAI*, pages 132–139, 2002.

[6] Ad Feelders and Linda van der Gaag. Learning bayesian network parameters under order constraints. *International Journal of Approximate Reasoning*, pages 37–53, 2006.

[7] David Heckerman. A tutorial on learning with bayesian networks. *M. Jordan, editor, learning in Graphic Models*. MIT Press, Cambridge, MA, 1999.

[8] M. Jaeger. The ai&m procedure for learning from incomplete data. *UAI*, pages 225–232, 2006.

[9] J.Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, pages 213–244, 1997.

[10] H. W. Kuhn and A. W. Tucker. Nonlinear programming. *Proc. of the second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492.

[11] Radu Stefan Niculescu, Tom M. Mitchell, and R. Bharat Rao. A theoretical framework for learning bayesian networks with parameter inequality constraints. *IJCAI*, 2007.

[12] F. Wittig and A. Jameson. Exploiting qualitative knowledge in the learning of conditional probabilities of bayesian networks. *UAI*, pages 644–652, 2000.